UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

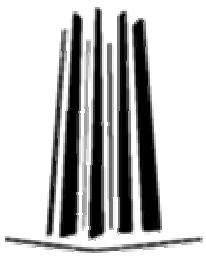
LA ARQUITECTURA JAVA EN LA CONSTRUCCIÓN DE APLICACIONES WEB, DIRIGIDAS A LA VENTA Y RESERVACIÓN DE BOLETOS DE AUTOBÚS

T E S I S

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

Asesor: César Francisco German Rosas



Presenta

Montes Samayoa Newman Edmundo

2007

AGRADECIMIENTOS

A los Dueños del equipo que lo crearon y están desde los palcos observando,

a la única Dueña del equipo que nos acompaña desde el terreno de juego,

al Head Coach que con su estrategia depurada guió al equipo hasta el juego de campeonato,

a la Quarterback que ejecutó todas las jugadas para llevarnos hasta la zona final,

y desde el backfield, a la Corredora de Poder que bloqueando en campo abierto permitió hacer las jugadas grandes,

a los Montes y los Samayoa,

a la UNAM y los Universitarios que me acompañaron,

a los Centuriones.

Gracias

ÍNDICE

OBJETIVO

El trabajo que a continuación se desarrolla tiene su razón de ser en la idea de introducir a los futuros desarrolladores de software al mundo del comercio electrónico, las aplicaciones web y su entorno práctico dirigido al lenguaje Java cuando éste se orienta hacia el desarrollo de sitios Web.

Establecer a manera de guía aquellos elementos teóricos y su relación con los elementos prácticos, enfocados a la construcción de páginas Web, mediante el uso del lenguaje de programación Java.

Construir parte de una aplicación que solucione una situación real a partir de los elementos estudiados.

JUSTIFICACIÓN

La motivación de encausar esta investigación hacia los objetivos planteados radica en mi propia vivencia, que ha sido breve, pero que tuve que encarar al momento de solucionar un problema real, en mi caso particular, durante mi servicio social aprendí parte de lo que son las aplicaciones Web haciendo uso de PHP y empezaba a conocer otros lenguajes como Java en aplicaciones sencillas. Sin embargo, al momento de enlazar los conocimientos que obtuve en mi formación universitaria con el uso de herramientas prácticas, como las que integran la plataforma Java, enfrenté muchos problemas para resolver esas situaciones debido a que no lograba concatenar dichos conocimientos con la práctica, es por esta razón que deseo aportar una guía para aquellos que enfrenten una situación similar a la que yo experimenté.

INTRODUCCIÓN

La presente investigación está dirigida para los casos donde los desarrolladores ya han programado en Java, pero no precisamente una aplicación Web, porque primero deben entender como funcionan las JSPs y los Servlets, y también para los casos donde ya se ha trabajado en aplicaciones Web pero en otras plataformas.

Cuando dejamos el mundo teórico para introducirnos a uno netamente práctico tenemos el conflicto de cómo verter dichos conocimientos en la práctica. A partir de dicho conflicto planteamos la siguiente hipótesis: Para llevar a cabo la construcción de una aplicación bajo la plataforma Java que permita comercializar a través de Internet, se requiere contar con una guía que permita enlazar lo teórico con lo práctico mediante el uso de los componentes propios de Java implementados a través de patrones de diseño, los cuales proponen una estructura estándar, de tal suerte que, bastará con visualizar un ejemplo que realice una tarea similar para tomarla como base y a partir de ésta generar una adecuada a las necesidades de nuestro caso particular.

En virtud de los elementos que se han expuesto, la presente investigación pretende seguir el camino planteado por la hipótesis, tomando el caso específico de las aplicaciones encaminadas a funcionar a través de las redes computacionales y la construcción de éstas bajo la plataforma Java.

Iniciaremos esta investigación en el apartado que denominaremos Antecedentes, explicando en que consiste la revolución en las comunicaciones, lo cual da fundamento a una nueva manera de comerciar haciendo uso de estos nuevos canales de comunicación.

El Capítulo Inicial explicará más a fondo lo que implica el Comercio Electrónico sus repercusiones y modalidades, para que nos sea posible visualizar lo que se refiere a esta innovadora manera de ejercer el comercio en la actualidad, lo cual dará pie para que en el siguiente capítulo expliquemos las particularidades que trae consigo el comercio electrónico en la construcción de las aplicaciones que permitirán dar vida a esta modalidad de comercio.

Entrando en la zona de la práctica entraremos al mundo de la construcción de aplicaciones, haciendo uso de los elementos que nos ofrece la plataforma JAVA. En el capítulo 3 explicaremos lo referente a esta plataforma y los elementos que la componen; en el capítulo 4 proponemos un caso práctico mediante el cual podremos aplicar la tecnología que habrá quedado explicada, para concluir con el capítulo 5 explicando de manera práctica el uso de la tecnología Java en la construcción de la aplicación propuesta.

ANTECEDENTES

A) La Revolución De Las Comunicaciones

La información es la forma en la que podemos plasmar y transmitir ideas o datos que nos ayudan a generar conocimiento.

El nacimiento de la red[1] de redes (INTERNET) dio paso a una nueva era en las comunicaciones permitiendo acortar las distancias y tiempos en los que viaja la información.

> Internet ha sido desde su inicio ante todo un medio para divulgar información, y como tal, una herramienta libre muy poderosa para el intercambio de conocimientos entre personas de todo el mundo[2]

[1] Cuando un conjunto de computadoras pueden compartir información entre si se dice que están conectadas en RED, aunque en la actualidad con los sistemas inalámbricos la conexión no es física sino virtual.
[2] http://www.htmlweb.net/manual/texto/texto_1.html

Para que este intercambio fuera posible fue necesario establecer un protocolo, o lo que es lo mismo un conjunto de normas y procedimientos útiles para la transmisión de datos, éstas debían ser conocidas por el emisor y el receptor para permitir de ésta manera la comunicación entre ellos y en consecuencia permitir la interpretación de la información que viajaba por la red, con el fin de que las computadoras entre las cuales se establece la comunicación puedan enviar la información a través de la red y las computadoras que solicitan la información puedan traducir la información de manera integra y veloz.

> En los primeros tiempos de Internet la transmisión de documentos mediante el protocolo HTTP tenía como única misión el permitir a las universidades (y a los militares americanos) poder mantener una comunicación simple entre ellas...[3]

El protocolo utilizado con mayor frecuencia en la actualidad es el Protocolo de Transferencia de HiperTexto[4] mejor conocido como HTTP[5] por sus siglas en inglés. Esto quiere decir que la información que se enviará será en forma de Hipertexto haciendo uso en este caso del protocolo HTTP.

[3] Ibíd.
[4] Hipertexto se refiere a cualquier texto disponible en la red que contenga enlaces con otros documentos. Utilizar el hipertexto es una manera de presentar información en la cual texto, sonido, imágenes y acciones están enlazadas entre sí, de manera que se pueda pasar de una a otra en el orden que se desee.
[5] HTTP HyperText Transfer Protocol o Protocolo de Transferencia de HiperTexto.

El siguiente problema al que se enfrenta es cómo convertir un documento determinado en Hipertexto. Para resolver esta situación surge un lenguaje de programación en forma de Hipertexto mejor conocido como HTML[6] y sus siglas se refieren a Lenguaje de Marcas de Hipertexto. Este lenguaje le dará a la información la forma de Hipertexto para que ésta pueda ser enviada mediante HTTP.

Por último el equipo de cómputo que recibe la información en forma de Hipertexto necesita traducir esta información para que el usuario pueda visualizar esta información, y para ello se usa un programa conocido como explorador de Internet; el cual interpretara el Hipertexto dándole cierto orden para que la información pueda ser entendida por el usuario. A ésta información presentada por el explorador de Internet se le conoce como Página Web.

La importancia de HTML fue tal, debido a que en un inicio los recursos de los equipos y conexiones eran bastante limitados, lo cual limitaba el uso de las páginas Web a simples documentos de texto plano, mediante las cuales las instituciones educativas y las universidades, compartían conocimientos y experiencias de manera rápida.

[6] **HTML** (Hypertext Mark-up Language – Lenguaje de Marcas de Hipertexto),

Como sus siglas lo indican HTML es un lenguaje que utiliza marcas o etiquetas[7]. Estas marcas son parte del lenguaje de programación y permiten darle un orden deseado a la información ahí plasmada. Este lenguaje se ha vuelto cada vez más completo, ya que permite la utilización de imágenes, sonidos y secuencias de video en los documentos, añadiéndose gran cantidad de etiquetas sujetas a un patrón común, que dan la posibilidad de escribir textos complejos que se pueden ordenar de acuerdo a las necesidades de la información que contenga.

Gracias a esto, ahora se visualiza a una página Web no como un documento simple, un mero soporte de transmisión de datos, sino como documento basado en la aplicación de un diseño estudiado y muy elaborado dirigido a la transmisión de una información cada vez más visual.

> El mercado actual de las aplicaciones cliente-servidor basadas en Internet, la agresiva competencia entre empresas dedicadas al comercio en línea y los intereses propios de empresas de software dedicadas a la creación de aplicaciones para el diseño de páginas Web, ha llevado a la sociedad a necesitar cada vez más profesionales especializados en el manejo de multitud de herramientas capaces de construir sitios Web competitivos y funcionales, primando siempre por encima de todo un diseño actual y adaptado a cada época y ambiente en concreto.[8]

[7] Estas Etiquetas o Marcas son parte del lenguaje y todo elemento que se encuentre dentro de el es susceptible a ser ordenado de determinada forma. Esta forma es definida por las etiquetas. Por ejemplo si un texto se desea resaltar en **negritas** deberá ubicarse dicho texto entre las siguientes etiquetas TEXTO donde la etiqueta marca el inicio de la acción y la etiqueta el fin de ésta. Obteniendo como resultado en el explorador de Internet lo siguiente: **TEXTO**

[8] Ibíd.

Cuando hablamos de una aplicación en el lenguaje de la informática nos referimos a un programa de software, es decir, un conjunto de instrucciones definidas explícitamente que están encaminadas a ejecutar una tarea específica la cual será ejecutada por una computadora. Por consiguiente, una aplicación cliente-servidor es un programa de software que brinda un conjunto de ordenes que dictarán el funcionamiento de una red donde los elementos que la integran son los clientes (equipos de cómputo que solicitan información) quienes son proveídos de información a partir de un servidor (equipo de cómputo que provee servicios o información).

Las aplicaciones de este tipo tienen su razón de ser en la nueva tendencia de poder realizar ciertas operaciones de manera remota, es decir, poder realizar actividades que interactúen de manera inmediata a distancia, por ejemplo de un país a otro, accediendo únicamente a una computadora que pueda ser conectada en red con aquella terminal con la que deseamos interactuar, podría ser, por ejemplo, por medio de la Internet.

De lo anterior podemos apreciar que las posibilidades que Internet ha brindado al mundo de las comunicaciones actuales, le ofrece una mayor amplitud a todas y cada una de las disciplinas que se puedan o quieran unir a esta tecnología, en muchas formas, desde la simple recolección de documentación, susceptible de ser compartida por todo aquel que cuente con una computadora y acceso a Internet en cualquier parte del mundo, hasta transacciones económicas del más alto nivel.

Es en este punto donde realmente queremos centrar nuestra investigación, por principio de cuentas, Internet constituye una revolución en las comunicaciones, permitiendo que exista una ínterconectividad global, la cual ha modificado la forma en la que se realizan los negocios en todo el mundo.

B) Negocio Tradicional vs. Negocio Electrónico

Los negocios que se llevan de manera tradicional son aquellos a los que ya estamos habituados, por ejemplo, si asistimos a un centro comercial como un supermercado donde de manera directa, es decir, físicamente tomamos los productos que requerimos, los depositamos en un carrito y al final nos dirigimos a una caja donde una empleada se encarga de hacer la cuenta de los productos que vamos a comprar y posteriormente nos dice el monto total de nuestra compra, así que pagamos ya sea con efectivo o con tarjeta y la transacción se ha completado.

Como este negocio, muchos otros funcionan de manera similar, sin embargo regularmente tenemos que adaptarnos a los horarios de dichos lugares. Cómo serían las cosas si por principio de cuentas no tuviéramos que estar físicamente en ese lugar para adquirir determinado producto, en segundo lugar pudiéramos hacerlo cualquier día y a cualquier hora en la que nosotros deseáramos y por último que nos lo entregaran directamente en la comodidad de nuestro domicilio. En otros tiempos resultaría descabellado el sólo pensarlo, pero en la actualidad es toda una realidad y de hecho nos comenzamos a habituar a estas actividades. Y esto es gracias a la introducción de lo que son los negocios electrónicos los cuales no serían posibles sin la existencia de la red de redes mejor conocida como Internet.

Las ventajas que un negocio electrónico tiene en comparación con un negocio tradicional pueden ser apreciadas de acuerdo con los siguientes señalamientos.

- Ocurre una disminución de los costos
- Reducción de costos en abastecimiento
- Aumento en el volumen de transacciones del mercado
- La existencia de materia prima y productos se reduce a lo necesario optimizando de esta forma gastos y espacio de almacenamiento.
- Prevalece un mayor control y orden de las transacciones tanto en el interior como las que van o vienen desde el exterior de la empresa.

- Las barreras geográficas y deferencias horarias dejan de ser un problema.

- Disminuyen obstáculos de canales tradicionales de distribución.

Todo esto ha permitido que la economía mundial se lleve de una forma dinámica mediante los mecanismos propios de los negocios electrónicos.

Esta nueva forma de hacer los negocios está englobada en el concepto de negocios electrónicos o electronic business (e-business)[9].

[9] En el entorno informático donde el idioma en el que se encuentra toda la literatura es el inglés, donde llaman a todas las tecnologías electrónicas anteponiéndole una "e "que se refiere a electronic como en el caso de correo electrónico mejor conocido como e-mail.

Capítulo 1

EL COMERCIO ELECTRÓNICO Y LOS SITIOS DE VENTA EN INTERNET

Debido a que las aplicaciones Web se pueden acceder mediante sitios de Internet y estos sitios se rigen mediante los lineamientos propuestos por el comercio electrónico el cual forma parte de los negocios electrónicos se considera importante hacer una introducción a todos estos elementos ya que resultan ser el fundamento teórico respecto a nuestro objetivo práctico y por tanto se debe tener un conocimiento sólido al respecto.

1.1 Negocios Electrónicos

E-business propone un estilo para utilizar las nuevas técnicas y tecnologías en las formas de hacer negocios. Con lo anterior nos referimos a que debido a la gama de posibilidades que ofrecen las nuevas tecnologías relacionadas a los sistemas computacionales e informáticos, es necesario replantear todos los procesos propios de la empresa, debido a que, cuando una empresa entra a este nuevo mundo se debe dar cuenta que algunos de los procedimientos propios de su negocio se tornan o tienden a ser obsoletos y por tanto debe considerar esta situación para realizar una redefinición de estos.

Es peligroso entrar al mundo de los negocios electrónicos, porque aunque se está entrando a un terreno nuevo y vanguardista, resulta clave hacerlo de la manera más adecuada, debido a que va a cambiar de manera radical la forma de operar el negocio, y si las decisiones en torno a este cambio no son bien estudiadas, puede llevar a la empresa a su propia desintegración.

Esto que mencionamos no lo hacemos con el fin de ahuyentar a aquellos que están en vías de un cambio hacia estas tecnologías, pero deseamos hacer hincapié en la necesidad de realizar esta renovación o reestructuración, haciendo un análisis profundo de todos los procedimientos, debiendo contemplar hasta el más mínimo detalle y de manera cuidadosa porque de ello depende el futuro exitoso de la empresa.

1.2 Procesos de Negocio

Anteriormente hacemos referencia a los procedimientos queriéndonos referir a aquellos mecanismos o formas en que se realizan las operaciones que rigen el funcionamiento de la empresa. Para no dejar lugar a dudas y ser preciso y claro, en este ámbito el término que se debe utilizar es el de procesos de negocio y de esta forma entender exactamente a que nos referimos.

1.2.1 ¿Qué es un Proceso?

Un proceso lo podemos definir como una secuencia de pasos a través de algún medio que es desencadenado por alguna acción determinada y que entrega una respuesta específica.

1.2.2 ¿Qué es un Proceso de Negocio?

Un proceso de negocio es una colección de actividades previamente descritas que definen el funcionamiento o transformación de la información suministrada en determinado momento, el cual será lanzado en el instante en que éste reciba un estímulo específico, que al ser ejecutado tendrá entonces que responder generando una salida específica para un cliente o un mercado en particular.

Cuando hablamos ya sea de Procesos o bien de Procesos de Negocio obsérvese que están presentes los siguientes elementos:

- Objetivo
- Entradas Específicas (Inicio)
- Salidas Específicas (Fin)
- Recursos
- Secuencia de Actividades

En un proceso de negocio pueden estar implicadas diferentes entidades propias de la organización, como pueden ser personas o máquinas, que trabajan de manera conjunta para obtener un objetivo concreto.

> Un proceso de negocio es un tipo especial de proceso que describe, desde un punto de vista orientado al mercado, las actividades de una organización. El principal objetivo de los procesos de negocio es satisfacer necesidades de los clientes.[1]

[1] FISTEUS, Jesús Arias, *Modelado de procesos de negocio. Aplicación en entornos móviles* (Tesis Doctoral), Madrid 2002.

El motivo central de la visualización de los procesos de negocio como tales es tener la capacidad de identificarlos, para posteriormente hacer una descripción, análisis y diseño del proceso, esto con el fin de definir una manera para ejecutar dicho proceso haciendo uso de la tecnología, la cual, nos permitirá mejorar los resultados y tiempos de respuesta. Lo anterior es indispensable para llevar el funcionamiento de la empresa hacia mejores niveles de producción.

Una vez diseñado el proceso de negocio éste debe ser encaminado hacia la automatización, es decir, que de alguna manera el proceso se ejecute de forma mecánica y a la postre intercomunique a las diferentes entidades que lo conforman en estricto orden secuencial, de acuerdo a los lineamientos establecidos por la definición de proceso de negocio, a esta actividad se le conoce como *workflow*.

Para que un sistema de *workflow* tenga sentido deberán existir entidades tales como humanos y máquinas que interactúen en su funcionamiento mediante aplicaciones de software.

Los procesos de negocio existentes en una empresa para integrarse al esquema del e-business deben ser examinados, documentados y aprobados para su posterior adaptación o reconstrucción de acuerdo a los lineamientos que el e-business propone. A esta redefinición o re-ingeniería de los procesos, cuando está enfocada al e-business se le conoce propiamente como E–ingeniería[2].

1.3 Facetas del e-business

El e-business esta compuesto por diferentes facetas algunas de las cuales mencionaremos a continuación:

- Gestión de las Relaciones con los Clientes o CRM por sus siglas en inglés
- E-learning: Educación, capacitación, etc. aprovechando la Tecnología
- E-Commerce: Venta por Internet
- Gestión de la cadena de suministros o SCM por sus siglas en inglés
- E-Aprovisionamiento: Gestión Automatizada de Compras

[2] E-Ingeniería es el proceso de re-inventar la forma en que se realizan negocios electrónicos, desde la distribución de bienes o servicios, la colaboración y trabajo dentro de la compañía hasta la negociación y trato con los proveedores.

Como podemos apreciar, e-business es un concepto muy amplio el cual encierra todo un conjunto de tecnologías y técnicas, que de hecho no está del todo definido y que por el contrario sigue creciendo y redefiniéndose y que persigue crear mercados en línea; los cuales logren, según Josep Valor

> ...convertirse en redes de servicios e-business integradas, que pasen de ser meras plataformas transaccionales a centros de asesoramiento neutral. Así deberían ofrecer una amplia gama de servicios de negocio (logística, finanzas, selección de personal), servicios de comunidad (elaboración de contenidos, especialización en medios y servicios de TI (externalización, consultorías), considerando la colaboración entre las empresas como fundamental. [3]

La cita anterior puede resultar un tanto complicada si no se entiende de manera precisa lo que buscan los negocios electrónicos. E-business propone de manera concreta la utilización de los recursos técnicos y tecnológicos que provee toda la gama de recursos computacionales para obtener el mayor provecho posible de los procesos internos de ésta, de tal suerte que sí las demás empresas relacionadas directamente entre si siguen el mismo rumbo, llegue un momento en que sea posible compartir información útil para ambas, y así hacer más eficientes los procesos de retroalimentación que existen entre una y otra.

[3]VALOR, Josep. *et. al. Aprovechando las oportunidades del B2B* (Ponencia), 2003

Por ejemplo, una empresa de venta telefónica de boletos regularmente acepta la compra de boletos con tarjeta de crédito, ésta debe ser autorizada para hacer el cargo del monto por los boletos al cliente, entonces estamos en un punto en donde una empresa requiere de la respuesta de la organización bancaria. De hecho actualmente esta situación ya la resuelve el sistema automáticamente, haciendo la petición directamente al comunicarse con el sistema del banco el cual le entrega una respuesta inmediata. Si este mecanismo se establece tanto con los proveedores, como con los bancos y se extiende hasta los clientes, podemos acelerar los procesos de compra, venta, distribución y abastecimiento; entre otros procedimientos requeridos de acuerdo al tipo de negocio.

> Una de las claves de la fortaleza económica es la productividad, la cual se define como la cantidad de producto que se genera por cada hora de trabajo. Un aumento en la productividad genera una mejora en la economía sin que esto se traduzca en un aumento en los precios y salarios. Los economistas atribuyen el fuerte crecimiento de la productividad a la existencia de potentes computadoras y tecnologías de punta como Internet....[4]

[4] OELKERS, Dotty Boen, *Comercio Electrónico .Serie Business*, México, Internacional Thomson Editores, 2003 p. 13

Como vimos anteriormente e-business se puede manifestar de diversas formas, por ejemplo cuando la empresa desea capacitar a su personal puede recurrir al e-learning (aprendizaje electrónico); si lo que quiere es que sus procesos de abastecimiento mejoren automatizándolos entonces recurrirá al abastecimiento electrónico; y si lo que requiere es hacer eficientes los procesos de negocio de la empresa en el ámbito transaccional debe entonces utilizar lo que es el comercio electrónico (e-commerce).

En cuanto a lo planteado anteriormente referente al e-business es importante aclarar que la intención de esta investigación está encaminada únicamente en la parte de los negocios electrónicos que se enfocan en el manejo transaccional de las operaciones de venta a través de Internet y la construcción de los sitios que sirven a este propósito, siendo esto solamente una parte del e-business.

1.4 Comercio Electrónico

Una definición válida pero no muy útil para el sentido de esta investigación es concebir al comercio electrónico como el intercambio de información, bienes o servicios a través de algún medio electrónico cualquiera que éste sea.

El comercio electrónico, como debe ser entendido en el entorno informático actual, es aquel que establece el intercambio de bienes, servicios e información, el cual es posible mediante el uso de *sitios en Internet*[5].

El comercio electrónico como es concebido actualmente tiene su origen en 1991 cuando Internet entró de lleno al uso comercial convirtiéndose en el medio ideal para estas actividades en formato electrónico, ya que pueden ser mostrados, vendidos y entregados a través del mismo medio a los diversos clientes. Es simplemente un nuevo canal de distribución que puede ser aprovechado para aumentar la eficiencia, reducir los tiempos de entrega y disminuir los costos del bien o servicio en cuestión mediante páginas Web.

La comunidad informática engloba este intercambio dentro del concepto de comercio electrónico o electronic commerce (e-commerce), el cual ha tenido su más importante desarrollo en los últimos años gracias a el continuo desarrollo de la tecnología tanto de hardware como de software, lo que ha permitido superar ciertos límites que le impedían su crecimiento, sin embargo las herramientas tecnológicas para la construcción de estos sitios siguen mejorando para cubrir oportunamente los requerimientos del vertiginoso avance de estas tecnologías.

[5] Un sitio en Internet es una localidad dentro de la red informática que permite interactuar a los usuarios con los servicios que éste ofrece haciendo uso de páginas Web las cuales visualizan la información para el usuario que acceda a ella.

El comercio electrónico es el ámbito idóneo de compra para los consumidores que desean extraer las máximas ventajas del mercado interno. El consumidor sólo necesita acceder a una computadora con una conexión a Internet para que pueda hacer uso de los sitios en los cuales pueda comparar precios, productos y servicios entre miles de comerciantes.

> El comercio electrónico es una fuerza dinámica dentro de la economía de mundo entero. El uso de la Internet en la actividad comercial está revolucionando la forma en la que el mundo corporativo realiza negocios, además está aumentando el poder de la economía, ya que este medio logra comunicar a las grandes empresas con los nuevos líderes comerciales.[6]

El comercio electrónico desde nuestro particular punto de vista como ingenieros y desarrolladores de software, no es solamente lo que se hace mención en la cita anterior, es algo más profundo, o al menos así se debe entender, ya que el introducir una tecnología de este tipo implica una gran cantidad de consideraciones, tanto para el desarrollador como para la empresa.

El Comercio Electrónico se puede concebir de muchas formas, para los usuarios puede decirse que es únicamente un medio para realizar una consulta, compra o solicitud de algún producto o servicio, porque sólo pueden manipular lo que la página les presenta en la pantalla y por tanto desconocen todos los procesos que se desencadenan después de realizar alguna acción dentro de la página.

[6] OELKERS, op. cit., p. 5.

Por otro lado, todo proceso que es generado por la página es el producto del trabajo realizado por el equipo de desarrollo[7] avocado a esta tarea, y la última parte de esta cadena es la empresa o institución que desea establecer el servicio de comercio electrónico.

Para cada una de estas partes que integran el ciclo del comercio electrónico implica una serie de consideraciones:

1. El usuario o comprador debe visualizar la conveniencia de utilizar éste y no otro medio para realizar su compra.

2. Para el equipo de desarrollo es necesario establecer todas las consideraciones necesarias para construir el producto en las mejores condiciones, de acuerdo a los requerimientos exigidos por la empresa y a la vez considerar las necesidades del usuario final para que el sitio sea accesible, atractivo y seguro.

3. La empresa debe definir las mejores estrategias para entrar al mundo del comercio electrónico.

[7] Equipo de desarrollo se le conoce a un grupo integrado por analistas, diseñadores y programadores que llevarán a cabo la creación del sitio de Internet que permitirá llevar a cabo el comercio electrónico.

En realidad el interés de ésta investigación se centra en el segundo punto, sin embargo por el hecho de encontrarse justamente en la parte medular del ciclo, el equipo de desarrollo no debe perder de vista los objetivos y requerimientos planteados por la organización que lo solicita y además brindar al usuario final la certeza de que la seguridad de la información que éste provea sea propiamente manejada a lo largo de toda la transacción.

El comercio electrónico puede definirse como:

> Metodología moderna de negocios que permite a las organizaciones, comerciantes y clientes reducir costos mientras se mejora la calidad de los productos y servicios, así como incrementar la velocidad de entrega.[8]

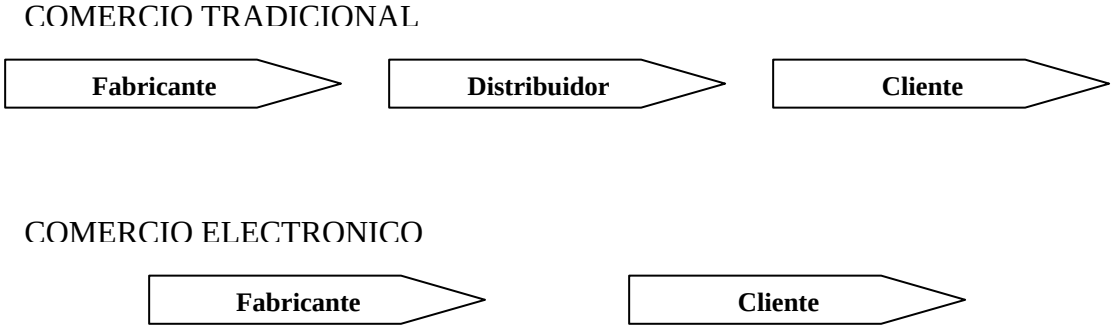
El comercio electrónico está generando una nueva forma de hacer negocios y con ello la forma en que los productos son distribuidos en el mercado, de tal suerte que los canales de distribución tradicionales se encuentran vulnerables a estos cambios, lo cual les puede afectar tanto positiva como negativamente.

[8] KALAKOTA, cit. pos. FRIAS, Jeann José, *Estudio y desarrollo de la seguridad en el comercio electrónico entre dos entidades productivas a través de Internet.* (Tesis de Maestría), México, 2000, p.15

Para entender lo anterior, imaginemos qué ocurriría con los distribuidores autorizados de automóviles si la marca que los provee, entra al mundo del comercio electrónico, de tal forma que los potenciales consumidores puedan tratar directamente con el fabricante para obtener lo que deseen, cuando lo deseen, incluso en la comodidad de su hogar y por si esto fuera poco, a un mejor precio, debido a que se estaría suprimiendo uno o más intermediarios, los cuales necesariamente incrementan el costo del producto.

Es evidente que lo anterior deriva necesariamente en conflictos entre los distribuidores y los fabricantes, obligando a los distribuidores a ejercer presión para que los cambios de este tipo no los afecten directamente.

En la siguiente imagen podemos apreciar como el comercio electrónico puede modificar los canales de distribución de productos.



1.4.1 Alteraciones en los canales de distribución

La introducción de una nueva tecnología, como lo es el comercio electrónico, da paso a un conjunto de efectos colaterales que pueden dar fin a cierto tipo de prácticas o empresas, y al mismo tiempo, ofrecen nuevas alternativas para formas innovadoras de realizar los negocios y por tanto abrir el mercado a nuevas empresas.

1.4.2 Desintermediación

En la actualidad para conseguir un producto de software es posible adquirirlo mediante la compra en línea del producto, directamente con el fabricante, quien posteriormente permitirá que el producto sea descargado de la red. Este tipo de operaciones provocará a aquellas distribuidoras de estos productos al menos una baja considerable en sus ventas, esto quizá no se aprecie inmediatamente, pero en la medida que este tipo de prácticas se generalice, será la medida en la que afecte a este tipo de intermediarios.

> Cuando un canal de distribución tradicional se vuelve obsoleto, éste desaparece. Este proceso de pérdida de canales de distribución se denomina desintermediación.[9]

[9] OELKERS, *op. cit.,* p. 12.

Para una empresa que se encuentra en este punto no todo está perdido, pero si resulta indispensable una redefinición en sus procesos de negocio, para ir cambiando en la medida en la que el comercio actual lo hace.

1.4.3 Reintermediación

El fenómeno opuesto a la desintermediación se le conoce como reintermediación y ocurre cuando un mecanismo de distribución se incorpora al sistema de distribución.

Podemos decir que el comercio electrónico es por sí mismo una nueva alternativa, que en los últimos años se ha incorporado al sistema comercial y que día a día adquiere mayor fuerza y propicia con ello que las prácticas tradicionales tiendan en ciertas circunstancias a desaparecer.

El comercio electrónico no termina definitivamente con los intermediarios y las prácticas de comercio tradicionales. Por un lado, ya que el consumidor requiere un producto, el vendedor debe hacerlo llegar al cliente y por lo regular el vendedor no lo hace directamente, éste contrata a una empresa de paquetería la cual se encarga de hacer llegar el producto. Por otro lado las prácticas de comercio tradicional siguen vigentes porque ciertos sectores de la población (que en nuestro país son la mayoría) no tienen acceso a los recursos, que en la actualidad son necesarios, para entrar al mundo del comercio electrónico. Pero estas son consideraciones que debe tomar en cuenta la empresa o institución con el propósito de acoplarse al mundo del comercio electrónico de la manera idónea, para que conviva con las formas del comercio tradicional y así obtenga el mayor provecho de ambas.

1.4.4 ¿Como funciona el comercio electrónico?

Utilizar Internet como una herramienta para facilitar las tareas del hombre ha dado paso a una nueva era. Una era donde las distancias se pueden recorrer en instantes, al menos de manera virtual.

Es real el hecho de que la información pueda viajar en instantes a otros países y continentes, pero también se puede estar en una reunión de trabajo de manera virtual, es decir, donde una computadora conectada a la red nos permita por medio de una cámara visualizar lo que sucede en la reunión y por el mismo medio que los asistentes a la reunión puedan escuchar lo que tenga que decir quien se encuentra en la otra terminal, esto se le conoce como conferencia en línea, así que el uso de la tecnología permite que el hombre se pueda ocupar de sus tareas a distancia usando la red de redes como se le conoce al Internet.

Cuando un sitio en Internet es construido para la venta y Reservación en Línea éste funciona de la siguiente forma:

1. El cliente que accede al sitio elaborado específicamente para la venta o reservación de un servicio (particularmente reservación o venta de boletos para viajar en autobús).

2. Los datos provenientes de un catálogo de viajes son mostrados al cliente mediante la página.

3. El cliente realiza una selección del catálogo para indicarle al sistema cual es el elemento que desea consultar.

4. El sistema le muestra información más detallada acerca del elemento seleccionado.

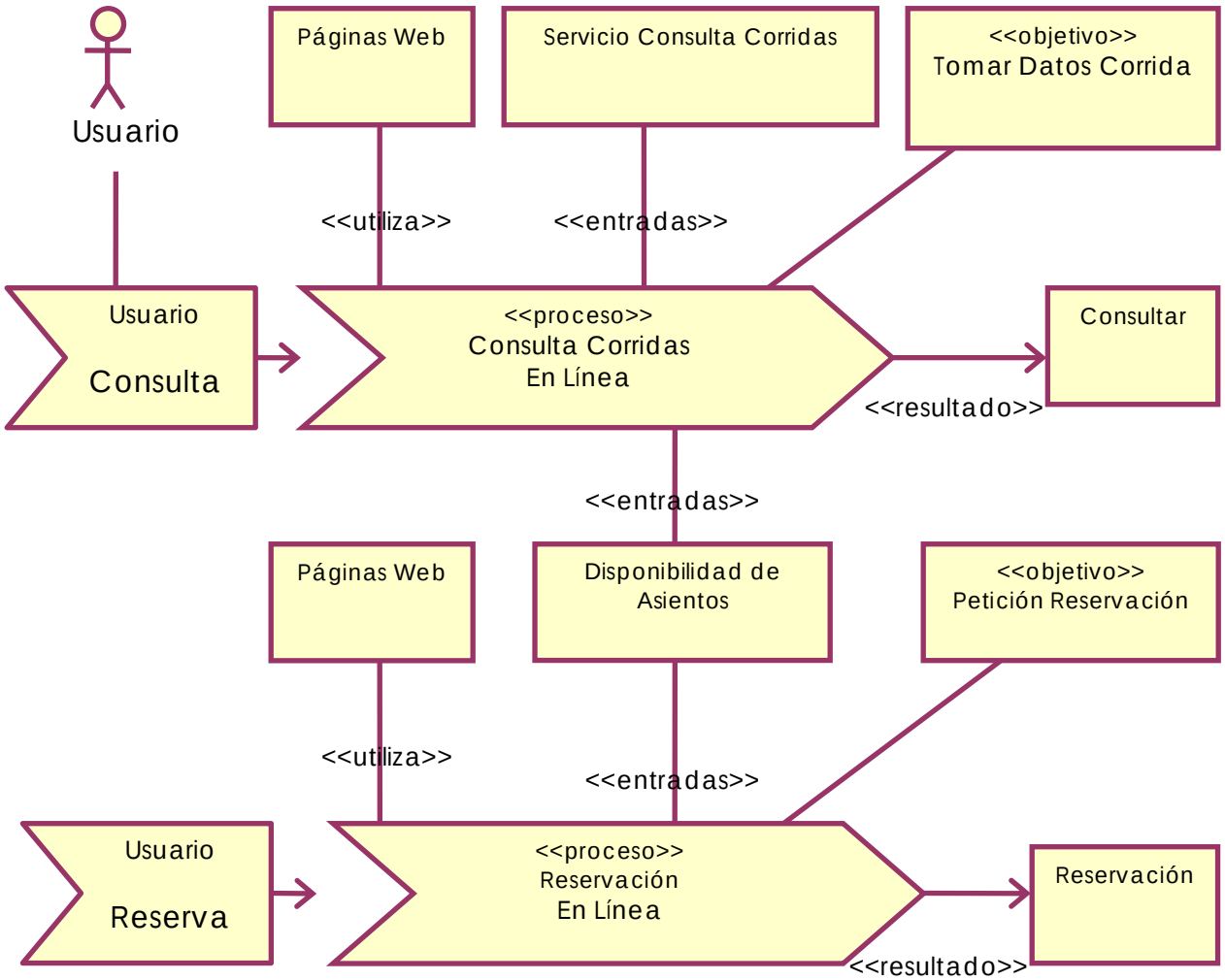
5. El cliente confirma que desea comprar el boleto, llenando un formulario que provee el mismo sitio, el cual recopila los datos del cliente y la tarjeta de crédito a la que se va a cargar la compra en el caso de comprar. En el caso de una reservación se suprimirán los datos concernientes a la tarjeta de crédito ya que se esta reservando un lugar quedando pendiente la compra de éste.

6. En cuanto el cliente envía sus datos, estos comienzan a procesarse, ya que el sitio necesita establecer un vínculo con la institución bancaria, la cual acreditará o desacreditará la tarjeta con la que el cliente planea pagar, y posteriormente en virtud de la respuesta que del banco al sitio se deberá comunicar el resultado al cliente.

7. Dicho resultado deberá quedar plasmado en un recibo que podrá imprimir el cliente.

Lo anterior se ha mencionado de manera particular para el caso planteado, pero si se desea aplicar en otro ámbito es deseable considerar las peculiaridades de cada negocio ya que al ofrecer productos y servicios diferentes es probable que algunos procesos se extiendan.

Este diagrama deberá contener la lógica de negocio correspondiente a la reservación de boletos para viajar en autobús.



El diagrama muestra dos procesos, Consulta Viajes y Reservación en línea, los cuales son generados por el Usuario, el medio de comunicación que **<<utiliza>>** es la Página Web. Estos procesos reciben como **<<entradas>>** consulta viajes y disponibilidad de asientos respectivamente como servicios provenientes de la empresa. La consulta de viajes es necesaria para que el cliente pueda elegir la que más le conviene y a partir de ella se le mostrara la disponibilidad de asientos, mediante la cual podrá elegir los asientos a reservar, si su petición es aprobada se ejecuta la reservación.

1.4.5 Procesos del Comercio Electrónico

Resulta indispensable conectar las diferentes etapas del comercio electrónico desde la inicial donde el consumidor pide el producto, hasta la última donde el proveedor verifica la viabilidad de la entrega, ya que se han dado casos en los que la solicitud rebasa los recursos previstos por la empresa y al ser un sistema automatizado puede ocurrir que éste permita peticiones aún cuando ya no se pueda proveer el producto o servicio, lo cual derivará necesariamente en la inconformidad del cliente, la que seguramente se verá reflejada en demandas legales. Es decir, que si existe la posibilidad de que la organización no tenga la capacidad de responder a las peticiones del cliente sea por falta de tiempo, materia prima, recursos humanos, etc., entonces se estará incurriendo en un grave error que traerá como consecuencia situaciones legales incómodas para la empresa.

La siguiente es una representación de los procesos que deben de integrar el comercio electrónico según IBM.[10]

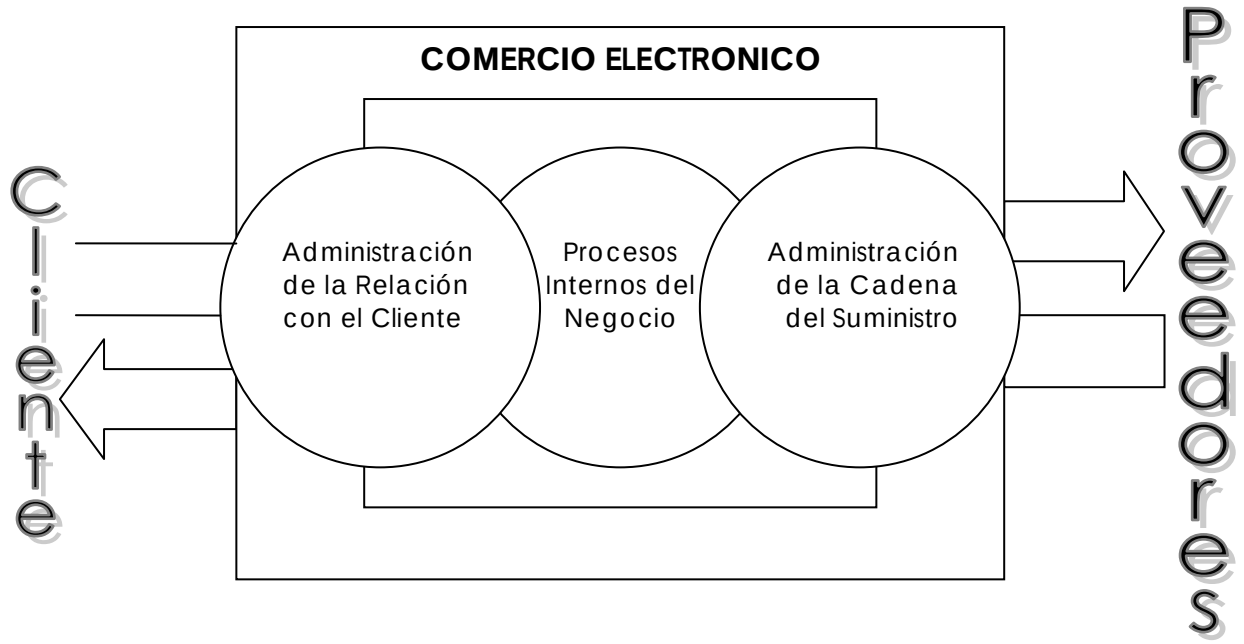


FIGURA. 1 Procesos de e-commerce IBM[11]

En la figura anterior podemos apreciar que las tres fases son interdependientes, es decir, que su funcionamiento conjunto es indispensable y no es admisible el hecho de que alguno de ellos trabaje sin la comunicación o interrelación requerida por las otras.

[10] IBM Application Framework for e-business, 2000
http://www-4.ibm.com/software/ebusiness/AppServices.html
[11] IBM Application Framework for e-business, 2000
http://www-4.ibm.com/software/ebusiness/AppServices.html

Quedando claro lo indispensable que resulta la intercomunicación entre los procesos, la parte más delicada de todo esto es sin duda el proceso de administración en la cadena de suministros, ya que requiere que las cadenas de suministro entre empresas trabajen en armonía para acelerar los procesos, reducir los costos y tiempos de entrega, y esto hace indispensable un nivel de seguridad adecuado, en el que todas las partes estén de acuerdo para que exista un apropiada regulación en las operaciones. Pues una empresa no va a permitir que la introducción de nuevas tecnologías hagan vulnerable a su empresa acarreándole más problemas que beneficios, así que es necesario asegurar la privacidad e integridad de la información, así como ubicar a aquellas entidades que participan en determinada operación concerniente a esta etapa mediante un registro de operaciones.

1.4.6 Modelos De Consumo Del Comercio Electrónico

Los modelos de consumo del comercio electrónico se clasifican de acuerdo a las entidades entre las cuales se dé, es decir, de quien lo provee y a quien está dirigido. Esta clasificación puede variar dependiendo del autor pero podríamos hablar de los siguientes:

- Consumidor a Consumidor (C2C - Consumer to Consumer)
- Consumidor a Negocio (C2B - Consumer to Business)
- Negocio a Consumidor (B2C - Business to Consumer)

- Negocio a Empleado (B2E - Business to Employee)
- Negocio a Gobierno (B2G - Business to Government)
- Negocio a Negocio (B2B - Business to Business)

1.4.6.1 Consumidor a Consumidor (C2C)

Consumer to consumer, es la denominación que se da a lugares de intercambio de información o servicios entre consumidores donde no interviene en momento alguno una empresa.

Si bien en los temas anteriores habíamos hablado de el uso del comercio electrónico enfocado al mejoramiento de los procesos empresariales, esto no necesariamente excluye a los individuos que desean comprar y vender sus productos directamente entre si.

Existen sitios específicos para realizar este tipo de modalidad en el comercio electrónico aunque hay quienes comparan esta modalidad con una venta de "garaje electrónico" debido a que típicamente se da la compra/venta de productos usados.

Algunos sitios dedicados a este tipo de comercio como lo es eBay también organizan subastas para los coleccionistas.

Realmente la conveniencia de esta modalidad radica en encontrar un producto dentro de una amplia gama de opciones y precios que, de entrada, son menores a los precios de los productos nuevos o bien es posible encontrar productos que típicamente no están a la venta en algún establecimiento en particular, sino que, son piezas que sólo un coleccionista puede valorar.

1.4.6.2 Consumidor a Negocio (C2B)

Esta modalidad lo que propone es que el consumidor sea quien ponga el precio a pagar por el producto.

Como sabemos esto sólo ocurre en los tianguis donde puede haber cierto "regateo" para adquirir lo que deseamos, de manera tal que el producto sólo se adquiere si ambas partes llegan a un acuerdo con el precio.

Una situación como la antes descrita no es siquiera imaginable en un centro comercial debido a que en estos lugares los productos ya tienen un precio definido e inalterable, sea una oferta o no, el precio al que se ofrece no está a consideración del cliente, éste cuando mucho podrá decidir si lo compra o no.

En la modalidad C2B, en realidad se maneja con lo que se conoce como modelo de subasta inversa (reverse-auction model), debido a que los consumidores hacen una oferta para adquirir un producto y la empresa está en libertad de aceptarlo o no. Esto puede parecer insólito, sin embargo esta situación funciona es ciertos casos específicos donde el producto o servicio se espera que no va a ser vendido en su totalidad, entonces resultará mejor venderlo a un precio inferior de su valor, ya que es mejor obtener algo a que se convierta en una pérdida total.

Para ver más claro lo anterior, pensemos en la venta de boletos para viajar en autobús, fuera de lo que son las temporadas vacacionales, resulta difícil que todos los viajes de todos los destinos posibles sean completamente vendidas, existirán en la mayoría de los casos lugares sin vender y si la empresa prefiere ocupar esos lugares en vez de dejarlos vacíos a expensas de vender el boleto por debajo de su valor está en libertad de hacerlo, y para ello es que, esta modalidad tiene su razón de ser.

Aún cuando esta modalidad en realidad no es muy utilizada en nuestro país es factible su uso, al menos debe ser considerada por el negocio, ya que en ocasiones se ignora que este tipo de alternativas existe.

1.4.6.3 Negocio a Consumidor (B2C)

Se le denomina comercio electrónico negocio a consumidor cuando una empresa realiza las funciones de mercadeo, promoción, publicación de información y catálogos de productos o servicios dirigidos a usuarios finales; mediante el pago con tarjeta de crédito o débito o cualquier otra forma de pago que admita el vendedor.

El sistema está dirigido al usuario final o consumidor y el procedimiento normalmente se realiza mediante la navegación de páginas en Internet con algún programa que permita el intercambio de información de forma segura.

Podemos visualizar que el comercio electrónico beneficia al consumidor de la siguiente manera:

- Intercomunicación hacia otros consumidores haciendo uso del correo electrónico, foros, videoconferencias y los grupos de noticias.

- La empresa puede ofrecer determinados servicios para la administración de las finanzas personales proporcionando programas de software especializados para administrar las inversiones, cuentas personales y el uso de herramientas de banca en su casa.

- El consumidor podrá adquirir productos, servicios o información en línea.

Para una transacción de comercio electrónico entre una persona y una empresa es necesario verificar la identidad de las partes involucradas, garantizar la privacidad e integridad de la información, es por ello que se utilizan mecanismos para codificar la información enviada por un usuario y evitar así riesgos y posibles ataques a la información. Normalmente se utilizan páginas codificadas con protocolos como el *Socket Secure Layer (SSL)* y el *Secure Electronic Transaction (SET)*.

Debido a las implicaciones anteriores, la seguridad es una de las principales preocupaciones de los usuarios del comercio electrónico persona a negocio, la confianza es un elemento tan importante que puede determinar el futuro y éxito de la empresa. En algunos países, como México, aún no se encuentra arraigada la confianza para el uso de los sistemas de comercio digital.

1.4.6.4 Empresa a Empleado (Business to Employee, B2E)

B2E es la sigla de business to employee (empresa a empleado). Se refiere a los procedimientos dirigidos hacia recursos humanos de la empresa a través de la Internet. Actualmente se aplica a intranets[12] en que los empleados de una compañía pueden obtener información acerca de la situación de la empresa como noticias, iniciativas o procedimientos propios de ésta para facilitar y en consecuencia agilizar las labores del empleado.

> El B2E trata de rentabilizar al máximo la eficiencia y el rendimiento del empleador, reduciendo al mismo tiempo la complejidad de sus tareas diarias. Los empleados son activos críticos de la empresa, y la mejora en sus relaciones con la empresa es un factor de vital importancia para la productividad global. [13]

Esta modalidad tiene la intención de abarcar todo aquello que la compañía puede hacer para conducir al personal hacia las más recientes y eficientes técnicas de mercado, ofreciéndole: beneficios, oportunidades de formación, horarios flexibles, bonos o participación en acciones y estrategias de capacitación de los empleados.

Para que esto sea posible es necesario construir un portal para empleados "B2E Portal",

[12] Intranet es una red local que utiliza los protocolos de Internet para comunicarse internamente en una institución.
[13] http://www.padsoft.net/internet/comercio-electronico/tipos.asp

> ...que es una página Web personalizada disponible para cualquier empleado de la organización. El portal B2E se centra en el individuo, siendo diseñado para incluir no solamente aquello que el empleado quiere encontrar en la Intranet (como es un directorio de empleados, información de soporte, etc.) sino también cualquier información personal y enlaces que el empleado quiera almacenar (como por ejemplo las cotizaciones de bolsa). La intención del portal no es solamente incrementar la eficiencia del empleado sino también su satisfacción y mejorar el sentido de "comunidad" dentro de la organización. [14]

Lo importante de este portal es que debe incluir de manera ideal ciertos elementos que le faciliten y atraigan al empleado para hacer de éste una herramienta asidua en su mecánica laboral. Por ejemplo:

- Fácil ingreso al portal mediante una dirección sencilla o accesible para acceder a la información, los recursos de la empresa, así como interactuar con los proveedores, los clientes y otros empleados.

- Publicación de desplegados y comunicados que informen al empleado las nuevas disposiciones de la empresa, además de compartir documentación que puede ayudar al empleado a desempeñarse mejor en sus tareas.

- La posibilidad de hacer el entorno más atractivo al empleado, radica en la posibilidad de hacerlo configurable de acuerdo con la comodidad y orden que el empleado requiera.

[14] http://www.deister.es/deisterwww/products/e-business/b2e/

Para que lo anterior sea posible, el portal debe contar con una infraestructura tecnológica capaz de gestionar la seguridad y el control de acceso a los recursos de una forma adecuada para evitar que la información se corrompa o sea utilizada de manera indebida o irresponsable.

1.4.6.5 Negocio a Gobierno (Business to Government, B2G)

La sigla de business to Government. Se aplica a sitios o portales especializados en la relación con la administración pública. En ellos las instituciones oficiales (ayuntamientos, diputaciones...) pueden ponerse en contacto con sus proveedores, y estos pueden agrupar ofertas o servicios.

Esta modalidad está orientada al intercambio de productos y servicios entre los proveedores y las Administraciones Públicas. Se trata de crear portales que actúen como plataforma de contacto entre los distintos entes de la Administración y las empresas suministradoras de bienes y servicios, para la realización de transacciones de comercio electrónico B2G.

1.4.6.6 Comercio electrónico Negocio a Negocio (B2B)

El comercio negocio a negocio permite a dos empresas realizar transacciones seguras, las cuales intercambian información directamente de sus sistemas computacionales. Anteriormente el intercambio de información se realizaba mediante el uso de redes privadas, pero en la actualidad esto se logra mediante el uso de las Extranets[15], las cuales permiten ligar las estrategias de intranets e Internet en el nuevo concepto de comercio electrónico. Así las órdenes de compra o pedidos, formatos de entrega, facturación y cobranza pueden ser enlazados para optimizar tiempos y costos.

Desde la perspectiva de Jeann José Frias Garza en su tesis "Estudio y desarrollo de la seguridad en el comercio electrónico entre dos entidades productivas a través de Internet", el comercio electrónico negocio a negocio facilita las siguientes actividades del negocio:

- Administración de los suministros para reducir los tiempos y costos de procesamientos de compras.

- Administración del inventario para reducir los tiempos de orden, embarque y distribución. De igual forma se reducen y optimizan los niveles del inventario en sus máximos y mínimos adecuados.

[15] Extranet es una Intranet extendida, es una red virtual constituida por dos o más intranets, que conjuntas permiten a Internet transportar información entre sus terminales.

- Administración de la distribución para facilitar el envío de documentación de embarque como son notas de carga, ordenes de compra y manifiestos de reclamo entre otros, permitiendo una mayor precisión en la información presentada.

- Administración de los canales de distribución para diseminar la información de forma rápida y segura acerca de especificaciones técnicas o de pago.

- Administración del pago para el envío o recepción de pagos electrónicos entre los proveedores o distribuidores incrementando la velocidad y reducción de errores al realizar los pagos de facturas.

De acuerdo con lo mencionado anteriormente, podemos apreciar que el propósito del B2B es llevar la mayor cantidad de procesos hacia la automatización, con el fin de que sean tan precisos como sea necesario, además de que estos se lleven a cabo de manera inmediata y segura.

> Al garantizar la seguridad de la información, los sistemas interorganizacionales cambiarán la forma en que las empresas conducen sus relaciones de negocios o como utilizan el Internet para ganar una ventaja competitiva. [16]

[16] Riggins Frederick J cit. pos. FRIAS, Jeann José, *Estudio y desarrollo de la seguridad en el comercio electrónico entre dos entidades productivas a través de Internet*. (Tesis de Maestría), México, 2000, p.17

La necesidad de hacer del comercio electrónico una herramienta totalmente fiable es esencial, es decir, no es admisible algún filtro o el más nimio error, sobre todo en el ámbito del B2B, ya que cualquier error o alteración en la información puede tener serios daños para todas aquellas organizaciones involucradas. Entonces cuando hablamos de seguridad nos referimos a dos aspectos principalmente. Por principio de cuentas la información debe viajar sin la posibilidad de que alguien ajeno al proceso pueda interceptar la información para hacer uso indebido de ésta; y por otra parte, que la información llegue a su destino de manera integra, exactamente como salió de origen.

De acuerdo con esto puede existir relación entre las diversas modalidades, lo cual nos indica que una no excluya a otras, ya que para que el B2C pueda funcionar de una manera óptima requiere que el B2B ya esté en funcionamiento dentro de la empresa.

Como ya habíamos explicado anteriormente un portal que permite al consumidor adquirir algún producto requiere necesariamente validar su pago directamente con el banco y de ser necesario, pasar el pedido a la empresa de paquetería responsable de hacer la entrega; es por ello que hablamos del concepto de negocios electrónicos, porque es toda una forma nueva de manejar el negocio con implicaciones profundas en la definición de los procesos propios de la empresa.

Capítulo 2

LAS APLICACIONES WEB, SU FUNDAMENTO Y LA VENTA EN LÍNEA A PARTIR DE ELLAS.

Ya hablamos de la razón de ser de los negocios electrónicos, del comercio electrónico y sus diferentes facetas, y de la necesidad en la que se ven implicadas las organizaciones para estar a tono con el mundo actual, pero esto es a un nivel teórico, para llevar esto a un nivel práctico y concretar estas ideas, lo fundamental es contar con los dos componentes que son la base tecnológica de una **aplicación enfocada al comercio electrónico**: la primera es la infraestructura tecnológica y la segunda es la arquitectura tecnológica. Ambas deben de correlacionarse, ya que a través de ellas se manipularán los procesos de negocio que previamente se hayan definido.

2.1 Aplicaciones Web

Una aplicación enfocada al comercio electrónico es típicamente una aplicación Web.

Una aplicación en general es un programa de software destinado a resolver un problema específico, por otro lado una aplicación cliente-servidor como mencionamos al inicio de esta investigación es un programa que dicta el funcionamiento de ciertos elementos a través de una red, sin embargo cuando la aplicación cliente-servidor se desenvuelve en un ambiente como lo es Internet se le denomina aplicación Web o Web Application.

2.2 Infraestructura Tecnológica

La infraestructura tecnológica está compuesta por aquellos dispositivos encargados de ejecutar los procesos y tareas concernientes a las comunicaciones (redes), los dispositivos (hardware), los programas (software), que serán los elementos básicos donde se sustentará el funcionamiento de la organización, que entra en el mundo de los negocios electrónicos.

2.2.1 Redes

En cuanto a las comunicaciones, como ya mencionamos anteriormente, se requiere de Internet como medio para poder tener contacto tanto con los clientes como con las empresas, anteriormente se establecían redes privadas pero implicaban un costo económico muy elevado, sin embargo, gracias a los rápidos adelantos tecnológicos, en la actualidad es fácil adquirir una conexión de banda ancha[1] para Internet, esto permitirá atravesar fronteras geográficas y llevar el negocio a nuevos niveles.

Sólo existe un problema con el uso de Internet, como recurso en las comunicaciones, y es que éste no es un medio seguro para la transferencia de la información, la cual es vital para todo tipo de transacción económica, es decir, que la información que viaja por Internet es vulnerable tanto para ser alterada, como para ser interceptada y hacer uso ilícito de ella, esto deriva del hecho de que en el momento en que fue creado Internet sólo se visualizaba como un medio para compartir textos o informes; sin embargo, en cuanto al inconveniente de la inseguridad, existen soluciones que podemos implementar usando determinadas estrategias de seguridad.

[1] Banda Ancha es un tipo de conexión en red de alta velocidad, es decir que permite la transferencia de datos en grandes cantidades en tiempos cortos.

2.2.2 Hardware

Cuando hablamos de hardware nos referimos al equipo de cómputo y los dispositivos que lo complementan, como pueden ser impresoras, scanner, etc.; que a fin de cuentas, serán los elementos que se encargarán de obtener la información, procesarla, para posteriormente dirigirla o canalizarla a donde sea menester.

> El comercio electrónico es posible gracias al crecimiento del poder de procesamiento de las computadoras de escritorio que ejecutan las aplicaciones y a los servidores que proveen el acceso a las bases de datos y aplicaciones.[2]

La implementación de un sistema acoplado a los negocios electrónicos requerirá de una infraestructura de hardware potente, es decir, de equipo actual que cuente ya con altas velocidades de procesamiento y gran capacidad de almacenamiento. Por suerte, hoy en día, esto ya es una realidad y los equipos que cuentan con estas características ya se pueden adquirir fácilmente.

[2] FRIAS, Jeann José, *Estudio y desarrollo de la seguridad en el comercio electrónico entre dos entidades productivas a través de Internet.* (Tesis de Maestría), México, 2000, p.33

Lo anterior se hace necesario debido a la exigencia de los sistemas operativos[3] actuales y a las aplicaciones más recientes, que exigen cada vez, mayores recursos de los dispositivos. Además, al introducir a una empresa en el comercio electrónico esto se acentúa, debido al incremento en el volumen de las transacciones, que derivará necesariamente en tener que adquirir equipos de alto rendimiento, que tengan la capacidad de trabajar 24 horas al día los 7 días de la semana en la más alta demanda de operación, además de que sea susceptible de tolerar fallas.

Para este propósito en la actualidad existen dispositivos especialmente diseñados para soportar la demanda descrita y se les conoce como Servidores. Un servidor[4] como dispositivo de Hardware es una computadora que funge como contenedor de información a la cual se conectan otros equipos para extraer la información ahí contenida.

[3] Los sistemas computacionales necesitan un entorno básico para su funcionamiento, es decir, un conjunto programas que permitan hacer uso de los recursos básicos con los que cuenta un equipo de cómputo, a estos programas se les conoce como Sistemas operativos.

[4] El término *Servidor* se utiliza para hacer referencia al dispositivo físico, pero también para hacer referencia al Software que lo hace funcionar; de tal suerte que, puede causar confusión al usar el término fuera de contexto. Una computadora cualquiera que cuenta con este software instalado también se le considera como servidor. Por ello resulta técnicamente mejor decir que un Servidor es un conjunto de Hardware y Software que responde a los requerimientos de un cliente.

2.2.3 Software

Pasando al tema del software es necesario considerar que, para implementar un sistema de comercio electrónico como el que se ha estado planteando, es necesario hacerlo funcionar sobre un sistema operativo como puede ser: Windows de Microsoft, UNIX, Linux, etc., donde uno de los cuales deberá ser elegido, tomando en cuenta su desempeño, confiabilidad, seguridad o estabilidad que se requiera. Cada uno de los sistemas operativos presenta ventajas y desventajas que deben ser estudiadas y consideradas para que al ser conjugadas con el sistema de comercio electrónico estos puedan interactuar de la mejor manera buscando sacarle el mayor provecho a ambas.

También es necesario establecer bajo que arquitectura se va a sustentar el proyecto, sin embargo, este tópico será abordado más adelante por ser un tema de mayor importancia donde se requiere mayor detalle.

Por último la elección de un manejador de bases de datos debe hacerse de acuerdo a las cantidades de información y de procesamiento de transacciones que utilizará la aplicación. Los más conocidos son: DB2, Informix, Oracle y Sybase aunque existe una gama muy amplia de manejadores de bases de datos que pueden ser consideradas. Además es recomendable hacer uso de un servidor de transacciones, el cual nos ayudará a mejorar el rendimiento de las aplicaciones.

La misión de esta investigación esta enfocada a la creación de una aplicación orientada al comercio electrónico, y es por ello que lo anteriormente mencionado es importante. La aplicación que queremos construir debe estar siempre enfocada a aquellos elementos que estamos considerando a lo largo de este estudio.

2.3 Arquitectura Tecnológica

> Una arquitectura es un conjunto de reglas, definiciones, términos y modelos que se emplean para producir un producto. La arquitectura Cliente/Servidor agrupa conjuntos de elementos que efectúan procesos distribuidos y cómputo cooperativo.[5]

Internet trabaja bajo el modelo de Sistemas Distribuidos debido a que varios equipos están interconectados mediante la red y tienen la facultad de comunicarse y coordinar sus acciones mediante el paso de mensajes, efectuando tareas diferentes, y a su vez, haciendo uso de los recursos de otros equipos. La forma en que se comunican es mediante protocolos, para que la información que es enviada pueda ser comprendida por otros equipos que usen estos mismos protocolos.

[5] CÁRDENAS, Alfonso Araujo, *Sistemas Distribuidos Concepto y Características.* (Articulo), 2004.

Desde que se inició la era de las redes computacionales los modelos que establecen la operación de estos sistemas han evolucionado. Uno de los primeros modelos fue el centralizado, donde un conjunto de equipos de cómputo eran conectados en torno a una misma computadora, regularmente una Supercomputadora (Mainframe) en la cual se llevaba a cabo el procesamiento de la organización; con este modelo existía el inconveniente de tener una capacidad de procesamiento limitado y al presentarse una saturación en las líneas de comunicación inevitablemente el sistema se caía. Posteriormente se utilizó un modelo por Grupo de Servidores, en cierta forma seguía siendo centralizado y regularmente eran servidores de archivos o impresión, pero tenían el inconveniente de prolongar los periodos de transmisión, si manejaban volúmenes grandes de información o varias peticiones simultaneas de los clientes. Existieron modificaciones y extensiones de estos modelos hasta llegar al modelo Cliente - Servidor que es el utilizado más comúnmente en la actualidad.

2.3.1 Modelo Cliente – Servidor

Este modelo identifica dos entidades: la máquina que realiza una petición de servicio se le considera cliente y el equipo que proporciona el servicio es el servidor. Los servicios que puede proporcionar son: Ejecución de Programas, Acceso a Bases de datos y Dispositivos. Las velocidades de transmisión en este modelo dependen directamente del tipo de conexión que se establezca.

2.3.2 Arquitectura Cliente/Servidor

La Arquitectura Cliente/Servidor establece que el cliente invoca cierto tipo de servicio, de tal forma que del lado del servidor, los equipos de cómputo realicen su trabajo de manera conjunta para entregar un resultado. La arquitectura Cliente/Servidor tiene diferentes formas, a continuación mostraremos las más representativas.

2.3.2.1 Arquitectura Cliente/Servidor de Una capa

El cliente realiza una petición al servidor, el cual contendrá todo el funcionamiento de la aplicación, desde la presentación y lógica operativa de la aplicación, hasta las bases de datos. Esto presenta el inconveniente de que el procesamiento requerido por cada petición se tiene que ocupar de las tres fases (presentación, lógica de operación y acceso a base de datos) y al presentarse un número muy grande de peticiones, los tiempos de respuesta se pueden alargar tanto que el cliente no obtenga el resultado a tiempo.

2.3.2.2 Arquitectura Cliente/Servidor de Dos capas

Para solventar el problema expuesto por la arquitectura anterior, aquí el trabajo se reparte en dos equipos; cada una de las capas tiene un propósito específico, la capa inicial tiene la presentación de la aplicación y su correspondiente lógica de operación, y la segunda capa contiene la base de datos o los posibles servicios que pueda proporcionar.

Al ser repartidas las operaciones en 2 equipos, se aminora la carga de procesamiento y recursos del equipo, el cual podrá responder más prontamente y atender una mayor demanda de peticiones.

2.3.2.3 Arquitectura Cliente/Servidor de Tres capas

En esta a diferencia de la anterior la capa inicial sólo contiene la presentación de la aplicación, ya que la lógica operativa de la aplicación se ejecutará en otro equipo y la base de datos y/o servicios en la última capa o equipo. Al simplificar más las tareas de cada capa los resultados mejoran en el rendimiento de la aplicación y los tiempos de respuesta.

2.3.2.4 Arquitectura Cliente/Servidor n capas

Siguiendo las arquitecturas explicadas hace unos momentos es factible concebir una arquitectura multi-capas que puede contener n(cualquier) número de capas, las cuales dependerán de los servicios que se puedan proporcionar y en caso de existir varias bases de datos, podrán ubicarse en diferentes equipos o tener servidores de respaldo de las bases de datos, por ejemplo:

A continuación explicaremos cómo quedaría constituida una arquitectura de 5 capas:

1. Capa de cliente: Se le conoce como cliente al equipo que va a solicitar un servicio, de tal suerte que una interfase gráfica (página Web) permita interactuar con el sistema implementado.

2. Capa de presentación: Nos referimos al conjunto de elementos programados que presentarán la información a través de la interfase al cliente.

3. Capa de lógica de negocio: Anteriormente señalamos la lógica operacional de la aplicación refiriéndonos a aquellos elementos que definen el comportamiento de ésta. Groso modo eso es la lógica de negocio, donde quedarán plasmados los procesos que transformarán la información suministrada en acciones, es decir, que la información mostrada en la interfase es lógica de programa, desde que esa información toma la forma de datos y realiza llamadas a métodos (función de la capa cliente) se desencadena el funcionamiento de la aplicación (lógica de negocio).

4. Capa de integración: La intención de separar en capas es independizar cada fase en la construcción de los sistemas o aplicaciones, es por ello que en esta etapa se separa toda la lógica correspondiente al uso de la información, que se dirige o proviene de las bases de datos, con el fin de simplificar la complejidad de acceso hacia la siguiente capa.

5. Capa de sistemas de información: Esta se refiere a la etapa donde se encuentran las bases de datos y aquellos elementos relacionados con ellas.

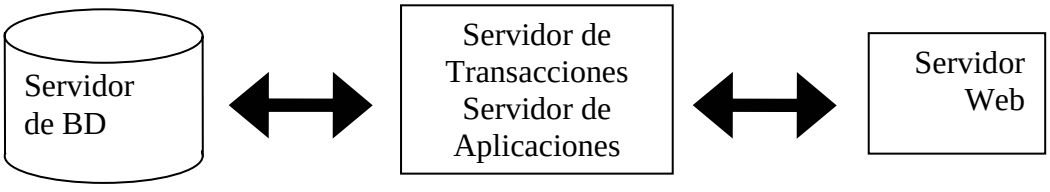
Lo que hace tan especial este tipo de modelos, es el hecho de simplificar cada etapa a una tarea específica para cada una de ellas, de tal forma que sean independientes entre sí; esto trae consigo diferentes beneficios, de entrada en este tipo de proyectos, existen grupos numerosos que se encargan del desarrollo, para que el trabajo pueda ser dividido entre los diferentes equipos que se dedicarán a la construcción de esta aplicación, sin que se entorpezcan unos con otros, ya que cada equipo sabrá con que información cuenta y que resultados debe entregar a la siguiente etapa, esto permite que los equipos desarrollen de manera paralela, es decir, que no es necesario que un equipo termine su parte para que el otro continúe con la respectiva, lo cual sería necesario si se trabajara bajo otro tipo de arquitectura. En consecuencia, un beneficio mucho más significativo es el hecho de que esta independencia y simplificación brinda cierta flexibilidad, que deriva en un acoplamiento[6] mínimo, es decir, que existe una interdependencia menor entre las capas permitiendo con esto realizar modificaciones cuando sea necesario, sin que esto afecte a las demás capas, trayendo consigo facilidades en cuanto a mantenimiento, extensibilidad y reutilización de código. Otro beneficio que con anterioridad habíamos mencionado es el hecho de que cada capa puede ser contenida por un equipo o servidor exclusivo, lo que permitirá mejorar el desempeño (*performance*) de la aplicación.

[6] Con el término acoplamiento se hace referencia al grado de interdependencia entre módulos (capas). El acoplamiento de estos puede ser considerado, dentro de una escala del más fuerte (el menos deseable) al más débil (el más deseable).

Una situación frecuente en el dinámico mundo de la tecnología ocurre cuando se originan nuevas tecnologías, las cuales en consecuencia traen consigo nuevos tipos de clientes, estas tecnologías deben ser consideradas y de ser posible realizar las adecuaciones necesarias para que sin importar la tecnología o dispositivos que se utilicen sea posible implementar una solución en la capa cliente y la capa de presentación sin que esto afecte al resto de las capas.

Lo que se busca al recurrir a alguna Arquitectura Tecnológica es obtener la mejor funcionalidad y desempeño de los elementos que ésta provee, es decir, tratar de encontrar la mejor forma de explotar los recursos tecnológicos, disponiéndolos en una estructura de trabajo acorde a las necesidades planteadas.

Veamos cómo se deben de disponer los componentes físicos de acuerdo a lo propuesto por la arquitectura tecnológica de IBM en su forma de una aplicación distribuida:



Esquema de Arquitectura tecnológica de IBM[7]

[7] Arquitectura Tecnológica (IBM), http://www-4.ibm.com/software/ebusiness/e-comServices.html

Como se puede apreciar el uso de sistemas distribuidos bajo el esquema anterior existen diferentes equipos desempeñando tareas específicas pero trabajando de manera conjunta, la pregunta obligada es ¿Cómo logran estos equipos trabajar de manera conjunta? No es tan simple como comprar los equipos y configúralos. Anteriormente usamos el término "aplicaciones" y es aquí donde reside la parte medular, es decir, la que incumbe directamente con el motivo de esta investigación y que indudablemente es la respuesta a la cuestión realizada.

Con las mejoras actuales en los aspectos de almacenamiento y procesamiento ya es posible delegar las exigencias de recursos a la tecnología misma, haciendo uso de técnicas de agrupamiento de servidores, esta técnica es conocida como **clustering**[8].

2.4 Sistemas Distribuidos

Son un conjunto de elementos computacionales autónomos que pueden ser heterogéneos, los cuales están físicamente distantes pero que pueden comunicarse entre sí haciendo uso de mensajes lanzados y recibidos a través de algún medio de comunicación.

[8] **Clustering** hace referencia al enlace de servidores individuales física y programáticamente y a la coordinación de la comunicación entre ellos, de modo que puedan realizar tareas comunes. Cualquiera de los servidores puede dejar de funcionar, y un proceso denominado de *failover* automáticamente conmuta la carga de trabajo a otro servidor para proporcionar un servicio continuo.
http://www.microsoft.com/spanish/MSDN/estudiantes/computadores/paralelas/clustering.asp

Aclaremos la definición anterior: concibiendo a un sistema distribuido como una manera de integrar un conjunto de equipos de cómputo a través de una red (medio de comunicación), en los cuales determinados protocolos, definen la forma en que se deben de componer e interpretar los mensajes enviados o recibidos, según sea el caso, permitiendo con esto que los equipos se puedan comunicar aún cuando cuenten con sistemas operativos distintos o pertenezcan a arquitecturas computacionales opuestas (heterogeneidad).

2.4.1 Características de los Sistemas Distribuidos.

Las características de los sistemas distribuidos pueden variar de acuerdo a las consideraciones en las que se enfoquen los diferentes autores, de la literatura correspondiente a este tema, sin embargo haremos mención de aquellas que para nuestro caso son las más útiles o que de alguna manera se relacionan con el propósito de la investigación.

1. La autonomía de los equipos permite independencia de plataforma (NT, Unix, Intel, RISC, Etc.).

2. Uso y control de recursos locales y remotos.

3. Medios de comunicación (Redes, Protocolos, Dispositivos, Etc.).

4. Capacidad de Procesamiento en paralelo.

Las aplicaciones enfocadas al comercio electrónico son conocidas como aplicaciones distribuidas, porque un elemento esencial en su funcionamiento es el hecho de que deben residir en un servidor de aplicaciones[9], el cual se recomienda sea instalado en un equipo autónomo, con el fin de que pueda atender y canalizar varias peticiones simultáneamente de todos los clientes que lo soliciten. De manera análoga los datos de toda una organización deben estar ubicados en un servidor exclusivo para la base de datos, para que ésta pueda atender varias peticiones rápidamente.

2.5 La Venta En Línea A Partir De Una Aplicación Web

Considerando los beneficios que las aplicaciones distribuidas proporcionan es menester guiar estas hacia lo que las aplicaciones Web proponen y abundar respecto a este tema enfocándolo a la venta en línea.

[9] Servidor de aplicaciones (Application Server) es una tecnología básica que proporciona la infraestructura que brindara los servicios y componentes necesarios para implementar un sistema basado en determinada plataforma.

Mediante Internet se tiene acceso a un sin número de páginas en las cuales se puede consultar información, proveerla, y entre muchas otras actividades se puede vender y comprar, para que esto último sea posible es necesario el uso de aplicaciones Web que son el medio por el cual se realizaran las operaciones de compra y venta, sin embargo esto, que se dice tan fácil, resulta ser una tarea muy compleja para crearla y requerirá de todo un equipo que se encargue de desarrollarlo. Este equipo deberá tener conocimientos respecto de lo que es el análisis, diseño, construcción e implementación de una aplicación de este tipo y aunque la gama de herramientas para realizar estas tareas es muy diversa, en los próximos capítulos explicaremos de manera sencilla lo que implica la construcción de una aplicación Web basándonos en la arquitectura Java que como veremos más detalladamente en el siguiente capítulo es una arquitectura de desarrollo en la cual se podrá construir el sistema.

Capítulo 3

LA ARQUITECTURA JAVA DIRIGIDA A LA CONSTRUCCIÓN DE APLICACIONES WEB

En primer término debemos saber que la empresa conocida como SUN Microsystems es la responsable del surgimiento del lenguaje de programación JAVA y algunos productos afines a este lenguaje los cuales constituyen lo que conocemos como Arquitectura JAVA.

Las plataformas tecnológicas basadas en este lenguaje son susceptibles de ser redefinidas, o bien, es posible crearlas de acuerdo a las necesidades que presenten las nuevas opciones tecnológicas, que como sabemos en el mundo actual suceden frecuentemente.

Para dar el paso a la creación de nuevas tecnologías, lo primero es realizar un documento donde se explique la necesidad de la tecnología propuesta y las razones por las cuales otras tecnologías no solucionan el problema existente; este documento constituye por si mismo una solicitud a la especificación de Java que es conocida por su nombre en inglés como *Java Specification Request* (JSR), la cual será considerada para su posterior aprobación en caso de que ésta se de. Si esto ocurre se crea una especificación, que es un documento en el cual se describe dicha tecnología haciendo referencia a las características que lo integran, así como la definición de sus partes y su interrelación. Finalmente se debe hacer una prueba de compatibilidad de la tecnología y una implementación de referencia.

3.1 JAVA

En un principio JAVA fue criticado por ser exclusiva de SUN Microsystems. Para poner fin a estas críticas JAVA decidió permitir la participación de otras organizaciones con el fin de que de manera conjunta se trabajara en la evolución JAVA y las plataformas basadas en la misma; esta decisión fue concretada el 8 de diciembre de 1998. Para realizar esto se creó un organismo integrado por aproximadamente 500 organizaciones, empresas, asociaciones y particulares, dicho organismo es conocido como Java Community Process (JCP).

3.2 Interfaz De Programación De Aplicaciones [API] Application Programming Interface

Un elemento esencial, el cual constituye una ayuda significativa para los desarrolladores es la interfase de programación de aplicaciones o ***API*** por sus siglas en inglés. Dicha interfase esta constituida por objetos (elementos informáticos programados que poseen identidad, estado y comportamiento) que simplifican cierto tipo de tareas las cuales están a disposición del programador para su uso, además de que proveen la documentación que le permitirá al programador visualizar para que sirven y como se utilizan.

3.2.1 ¿Qué es el API?

Es una interfase, la cual contiene un conjunto de especificaciones que brindan al programador elementos o rutinas predefinidas, con la finalidad de lograr determinado nivel de abstracción en la programación, debido a que proporciona cierto número de funciones de propósito general, que se encargan de lidiar con elementos comunes en el desarrollo, y de esta forma permite a los programadores concentrarse en situaciones propias de la solución (aplicación) y olvidarse de crear componentes de bajo nivel.

Es una interfase, porque se encarga de comunicar los componentes de bajo nivel con los elementos programados por los desarrolladores, valiéndose de la funcionalidad que el API les provee.

3.3 El API JDBC

El API JDBC es un API para la conectividad con sistemas de bases de datos relacionales. Este API esta compuesto de dos partes:

1. Una interfase a nivel aplicación usada por los componentes de la aplicación para acceder a la base de datos

2. Una interfase del proveedor que se sirve para adjuntar un controlador JDBC a la plataforma Java. Este controlador será proporcionado por el fabricante.

El API JDBC le permite ejecutar sentencias SQL desde los métodos programados en el lenguaje Java. El uso del API JDBC ocurre cuando se tiene un bean de sesión que accede a la base de datos. Es posible también hacer uso del API JDBC desde un Servlet o página JSP para acceder a bases de datos directamente sin embargo es una práctica que no se recomienda ya que se hacen vulnerables los elementos correspondientes a la lógica de negocio, para ello existen patrones de diseño que establecen una manera ordenada de trabajar haciendo la aplicación más segura, entre otras cosas, pero este tema (patrones de diseño) lo trataremos en un apartado exclusivo en este mismo capítulo.

3.3.1 Tipos De Controladores JDBC

Existen cuatro tipos de controladores JDBC:

1. Puente JDBC-ODBC
2. Java-Binario
3. 100% Java/Protocolo Nativo
4. 100% Java/Protocolo Independiente

3.3.1.1 PUENTE JDBC-ODBC

Este controlador convierte todas las llamadas JDBC en ODBC y realiza también la conversión correspondiente a sus resultados.

La mayoría de los controladores ODBC convierten a su vez a librerías nativas del fabricante por lo que es muy lento su desempeño.

Requiere de una instalación ODBC existente y configurada.

3.3.1.2 JAVA-BINARIO

Este controlador salta la capa ODBC y se comunica directamente con la librería nativa del fabricante de la Base de Datos. Es 100% Java pero necesita las librerías nativas de la BD en la máquina cliente.

3.3.1.3 100% JAVA / PROTOCOLO NATIVO

Escrito 100% en Java y es independiente de la máquina donde se va a ejecutar el programa.

Se comunica con la BD utilizando el protocolo de red nativo del servidor, la desventaja es que el cliente esta ligado a un Servidor de BD específico.

3.3.1.4 100% JAVA / PROTOCOLO INDEPENDIENTE

Escrito 100% en Java y es independiente de la máquina donde se va a ejecutar el programa.

No requiere configuración por parte del cliente. Utiliza un Protocolo de red independiente del Servidor de BD.

Proceso JDBC para interactuar con una base de datos.

- Selección del controlador
- Carga del controlador
- Obtención de la conexión
- Declaración de una sentencia SQL
- Ejecución de una sentencia SQL
- Resultado de la ejecución de una sentencia SQL
- Liberación de los recursos.

3.4 Contenedores

Típicamente en las aplicaciones multi-capa resulta difícil la programación donde se encontrarán miles de líneas de código utilizado para manejar ciertas transacciones y hasta cuestiones de manejo de recursos de bajo nivel. Existen productos que proveen servicios subyacentes que dan forma a lo que conocemos como contenedores, es decir, que un componente cuenta con determinada cantidad de servicios de bajo nivel que ya no será necesario programar, sino que el contenedor se encargará de este engorroso trabajo para que así el equipo de desarrollo se concentre en resolver directamente los problemas de negocio.

3.4.1 Servicios de Contenedores

Los contenedores son la interfase entre un componente y la funcionalidad específica de bajo nivel de la plataforma, dicha funcionalidad será la que soportará a los componentes.

Previamente a que algún componente sea ejecutado, éste debe ser ensamblado dentro de un módulo de la plataforma para su posterior despliegue o publicación en su contenedor.

El proceso de despliegue o publicación (deploy) al que nos referimos instala los componentes de una aplicación en los contenedores correspondientes.

3.4.2 Contenedor Web

La misión del contenedor Web radica en la administración de la aplicación Web, ya que es en ella donde será instalada la aplicación y además, es quien se encarga de ejecutar las clases Java en función de las peticiones HTTP correspondientes.

Para ser más claros, en este sentido un contenedor Web provee la infraestructura tecnológica necesaria para que pueda tener un funcionamiento adecuado nuestra aplicación. Esto ocurre porque cuando instalamos un contenedor Web en nuestro equipo, (sea de desarrollo o ya en producción) éste crea una estructura de directorios con los archivos instalados en el sistema para su funcionamiento. Dentro de esta estructura existe una carpeta designada para instalar las aplicaciones Web que en nuestro caso estaremos desarrollando.

El sentido de realizar lo anterior radica en que el contenedor Web es indispensable para proporcionar funcionamiento a determinados componentes de Java que para algunas especificaciones resultan indispensables, sin embargo, hablaremos de estos en los próximos temas.

Un contenedor Web tiene la funcionalidad de un servidor Web, el cual es capaz de recibir peticiones HTTP, y además permite ejecutar las clases de Java correspondientes a éstas, y por tanto vincular su funcionamiento con otras páginas HTML en respuesta, constituyendo así lo que llamamos una aplicación Web.

Algunos Servidores de aplicaciones contienen sus propios contenedores Web, sin embargo, es posible adquirir estos contenedores de manera separada, y en virtud de lo que los sistemas distribuidos proponen, se aconseja que el servidor de aplicaciones se ejecute en un equipo, y el contenedor Web en uno diferente, para aminorar la carga de trabajo que se daría si se utilizara un sólo equipo, pero sobre todo, para separar las capas de presentación y la capa de lógica de negocio.

Existen cantidad de contenedores Web, de diferentes marcas y potencialidades, sin embargo, la más conocida es Apache Tomcat que por el hecho de ser escrita totalmente bajo el lenguaje java y la cualidad de ser software libre, resulta conveniente para proyectos como el que se está planteando en este trabajo.

Los contenedores Web son la puerta de las aplicaciones Java a la World Wide Web.[1]

[1]World Wide Web, se refiere al sistema que permite el acceso al universo de información que incorpora al conocimiento humano.

3.4.2.1 Apache Tomcat

> Apache Tomcat es sin lugar a dudas el proyecto de software libre más famoso escrito en Java. Creado a partir de código donado por SUN Microsystems a la Apache Software Foundation, ha crecido hasta convertirse en la implementación oficial de referencia de Servlets y JSP sustituyendo a la implementación de SUN original. Esto lo convierte, obviamente, en el contenedor sobre el que todos los demás prueban su adhesión a las especificaciones.[2]

Por razones que pueden resultar obvias sabemos que puede integrarse sin demasiados problemas con el servidor Web Apache por ser de la misma compañía, y para aplicaciones empresariales grandes con servidores de aplicaciones como JBoss.

Sabemos que el uso de Tomcat a nivel mundial va más allá de su utilización en entornos de desarrollo libres como servidor de Servlets y JSP y que es utilizado en entornos de tipo comercial también.

Las ventajas de los servicios Web frente a sistemas de comunicación tradicionales son bastantes: independencia de la plataforma, independencia del lenguaje, uso de tecnologías y protocolos estándar, al tiempo que promueven el aislamiento de las diferentes capas de los sistemas empresariales, lo que redunda en una mayor extensibilidad, flexibilidad, etc. Tradicionalmente los servicios Web se basan en SOAP bajo HTTP como protocolo de transporte.

[2] MOLPECERES, Alberto Touris y PÉREZ, Martín Mariñán, *Arquitectura empresarial y software libre, J2EE,* España, javaHispano, 2002. p. 22

3.5 Componentes Java

Existen diversos componentes que forman parte de la JCP. Estas especificaciones tienen un objetivo definido y por tanto delinean la estructura que debe seguir un programa para ser considerado como un componente determinado.

Un componente es un elemento de software programado (programa), el cual sigue los lineamientos definidos por su especificación para que en conjunción con otros archivos, clases e inclusive otros componentes, se comuniquen entre sí y al ser ensamblados dentro de una aplicación den funcionalidad a ésta.

Los componentes son escritos en el leguaje de programación Java y son compilados de la misma forma que cualquier programa del lenguaje. La diferencia entre los componentes y las clases Standard de java es que los componentes están ensamblados de una manera acorde con la JCP y son JSRs que residen dentro de un contenedor el cual les provee la funcionalidad requerida.

Los componentes de Java son los siguientes:

- Java Servlet
- JavaServer Pages - componentes Web que se ejecutan en el servidor
- JavaBean
- Socket

3.5.1 Tecnología Java Servlet

Para poder entender que es un Servlet y cómo es que funciona, se debe tener claro que es una especie de puente o elemento intermedio entre las peticiones que realiza el cliente y la lógica de negocio, y en sentido inverso, entre las respuestas que la lógica de negocio entrega al cliente.[3]

Las peticiones más comúnmente utilizadas son las realizadas por HTTP que son las generadas por las páginas Web. Estas páginas están conformadas por un elemento <FORM> que llamaremos Formulario, el cual es el que reúne los campos y sus respectivos valores, los cuales componen una petición hecha por el cliente.

Esta petición es conformada de acuerdo a un método predefinido en la página, es decir, que la forma en que la petición será armada dependerá de la configuración del formulario. Para esto existen dos tipos, el método GET y el método POST.

- GET: Es el método en el cual los campos y sus valores forman parte de la URL[4].

- POST: Es el método en el que los campos y sus valores viajan en el cuerpo de la petición.

Una vez construida la petición ésta viaja mediante un protocolo de transferencia que es HTTP para este caso específico.

[3] A esta manera de proceder se le conoce como modelo de programación petición/respuesta o request/response, si hablamos en términos afines del lenguaje Java sin hacer uso de traducción.
[4] URL: Enlace único de referencia (Unique Reference Link) es decir un enlace identificado por una dirección de RED única la cual dirige al USUARIO hacia un SITIO en la RED.

La tecnología Java Servlet le permite definir clases tipo Servlet específicas para HTTP. Una clase Servlet extiende las capacidades de servidores que hospedan aplicaciones accedidas mediante el modelo de programación request-response. Aunque los Servlets pueden responder a cualquier tipo de petición, cuando se trabaja bajo ciertos lineamientos los Servlets son utilizados comúnmente para extender las aplicaciones hospedadas por los servidores Web

Un Servlet es un programa escrito en Java, por lo cual propiamente es una clase que se ejecuta del lado del servidor, acepta peticiones de clientes y genera respuestas dinámicamente mediante un protocolo de petición/respuesta. El tipo más común de los Servlets es un HttpServlet, acepta peticiones y genera respuestas HTTP.

Para que un Servlet pueda ser ejecutado es necesario que éste se encuentre hospedado en un contenedor Web en donde deberán cumplir con lo siguiente.

- Estar colocados en el directorio Web-INF/classes de la Aplicación Web.

- Ser registrados en el archivo descriptor de la Aplicación Web.

3.5.1.1 Procesamiento De Las Peticiones

El Método service() es un método que se comunica con la página Web, la cual se encarga de obtener la información recopilada por los métodos GET o POST provenientes de la página.

El método service() es utilizado para procesar las invocaciones a los métodos doGet (), doPost ().

- El método doGet() es invocado cuando se realiza una petición GET
- El método doPost() es invocado cuando se realiza una petición POST

3.5.2 Tecnologías Alternativas

Previo a la época de Java los servicios y peticiones que realizan los Servlets eran realizados por la tecnología de CGI que son la contraparte de los Servlets, sin embargo, los Servlets son más eficientes, adecuados, poderosos, transportables, seguros y económicos que los CGI.

3.6 Tecnología JavaServer Pages

Cuando los sistemas de información entran a la era de las redes e Internet, surge la necesidad de compartir información, y que la transferencia de ésta, sea de la manera más rápida posible, es por ello que surge HTML, que es el formato más liviano en tamaño, y en consecuencia el tiempo de transferencia es menor. HTML tiene su éxito en el hecho de que es un formato de texto plano, es decir, que está constituido única y exclusivamente por texto.

La tecnología JavaServer Pages (JSP) le permite al programador agregar porciones de código Servlet directamente en un documento tipo texto como puede ser una página HTML. Una página JSP también sigue siendo un documento de texto plano que contiene dos tipos de texto:

- Datos en estructura estática[5], los cuales pueden ser expresados en cualquier formato tipo texto como lo son HTML, WML, y XML.

- Componentes JSP, los cuales determinan la construcción del contenido dinámico[6] de la página.

De manera más clara podemos decir que una JSP es un documento HTML, con código Java inmerso entre código HTML. Donde el código Java es el que le otorga su funcionamiento dinámico.

Una JSP realiza las siguientes acciones:

[5] Una página Web tiene un comportamiento estático cuando todo proceso es manejado del mismo lado del cliente, por ejemplo cuando una página realiza una acción sin hacer una petición al servidor.
[6] Una página Web es dinámica cuando su comportamiento implica alguna transformación, producto de un proceso realizado por el servidor, el cual resuelve la petición realizada y la devuelve en una página nueva.

- Provee una respuesta dinámica basada en la petición desde el cliente

- Es portable.

- Se compila y ejecuta como un Servlet.

Aquí es donde podría surgir la pregunta de por qué utilizar esta tecnología y no otras, que posiblemente son más fáciles de utilizar y quizá más económicas.

La importancia de utilizar estos elementos propios de Java no es un capricho, sino que hay una gama muy amplia de razones por las cuales estos elementos son más poderosos que otros, como php o asp en el caso de los JSP.

Los JSP al estar diseñados bajo la plataforma de Java adoptan de ésta su transportabilidad y potencia.

3.6.1 ATRIBUTOS DE LAS JSPs

Config: Objeto de tipo ServletConfig, el cual representa la configuración inicial de las JSP

Out: Objeto tipo JspWriter que se utiliza para pintar contenido dinámico desde los Scriplets.

pageContext: Proporciona acceso a varias propiedades de la página. Es de tipo Page.

3.7 JavaBeans

Los JavaBeans son objetos java que siguen un patrón determinado (ver Transfer Object en la siguiente sección) el bean encapsula sus propiedades declarándolas privadas y provee métodos de acceso públicos (getter/setter) para leer (getter) e introducir o modificar (setter) los valores de estas propiedades.

3.8 Sockets

Un Socket es un componente que se utiliza para enlazar dos elementos para hacerlos compatibles, y por tanto en el mundo de Java resulta ser algo aplicable a esta definición tan simple.

En el ámbito de una arquitectura cliente-servidor no es indispensable el uso de Sockets ya que la comunicación de hecho se da. Sin embargo si queremos agregarle un elemento de seguridad al sistema que desarrollemos en Java será necesario usar los Sockets.

La definición de Java de un Socket de acuerdo con Sun Microsystems es la siguiente:

Un socket es una terminal de enlace en la comunicación de dos vías entre dos programas que corren en una red. Un Socket está asociado a un número de puerto de tal suerte que la capa TCP pueda identificar la aplicación a la cual será enviada la información.[7]

Un Socket de Java está definido mediante una clase, por tanto las llamaremos clases tipo Socket las cuales son utilizadas para representar la conexión entre los programas cliente y servidor. El paquete java.net provee dos tipos de clases, Socket y ServerSocket, la primera implementa la conexión del lado del cliente y la segunda conexión del lado del servidor.

Normalmente un servidor corre en una computadora específica y tiene un Socket el cual esta asignado a un puerto específico, El servidor sólo espera, atiende al Socket para que el cliente haga una petición de conexión. Por tanto los elementos que un Socket necesita para trabajar son el nombre del equipo en la red interna con la cual se va a comunicar y el número del puerto por el cual tendrá entrada y/o salida.

[7]"A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent." http://java.sun.com/docs/books/tutorial/networking/sockets/definition.html

Lo más importante en la razón de ser de los Sockets es que permiten ocultar aquellas particularidades que posee determinado sistema las cuales lo podrían hacer vulnerable, pero sobre todo su importancia radica en el hecho de que al hacer uso de la clase Socket en lugar de depender de código nativo los programas Java pueden comunicarse a través de la red en un estilo de independencia de plataforma.

3.9 Patrones de diseño

En la creación de software no existen lineamientos irrestrictos, que indiquen como se debe programar, únicamente debe atenerse a las reglas sintácticas del lenguaje, de tal suerte que el programador tiene la libertad de trabajar según le parezca conveniente.

Para evitar lo anterior existe un método conocido como: programación orientada a objetos, el cual establece una manera de programar haciendo uso de elementos informáticos programados que poseen identidad, estado y comportamiento (objetos), los cuales actúan de manera conjunta y organizada sin perder su independencia de funcionamiento.

La programación orientada a objetos provee flexibilidad, modularidad y reutilizabilidad del código al momento de desarrollar bajo este paradigma. De manera breve podemos decir que la programación orientada a objetos esta encaminada a desmenuzar el problema a resolver y con esto reducir la complejidad, esto se logra creando objetos. Estos objetos permiten adaptar el desarrollo de software a nuevas necesidades y/o requerimientos ya que al estar el problema fraccionado sólo se modifica aquel objeto relacionado directamente con el cambio y lo demás seguirá funcionando sin alteraciones, esto, como hemos dicho antes incrementa la flexibilidad de lo que se programa proporcionándole extensibilidad a nuestra tarea de programación.

La programación orientada a objetos puede incrementar sus potencialidades mencionadas anteriormente si se mejora la calidad del diseño y para esto existen los **patrones de diseño.**

El principal objetivo de los patrones de diseño es capturar buenas prácticas que nos permitan mejorar la calidad del diseño de sistemas, determinando objetos que soporten roles útiles en un contexto específico, encapsulando complejidad, y haciéndolo más flexible.[8]

Si bien es cierto que el uso de patrones no es forzoso, si provee una práctica deseable para los desarrolladores, ya que el conocerlos les permitirá entender la manera de atacar un problema para así resolverlo.

[8] DÍAZ, Moisés Daniel, *Diseño de aplicaciones Internet usando los Patrones de diseño J2EE.*

Como esta investigación esta dirigida a la Arquitectura Java y debemos saber que Java esta concebido para trabajar bajo el paradigma de programación orientada a objetos y respecto a los patrones de diseño Sun Microsystem establece una gama de patrones los cuales proveen soluciones precisas y documentadas a los problemas comunes de diseño de aplicaciones.

Un patrón de diseño surge cuando existe un escenario determinado en el diseño el cual debe ser atendido de manera específica. Para crear un patrón de diseño deben identificarse ciertos criterios que permitirán identificar el problema y darle solución tendiente a la estandarización, para que cuando ocurra esta situación determinada ya exista la manera de solventarla.

Para definir un patrón de diseño según Sun Microsystem se debe generar un documento basado en una plantilla que define las siguientes secciones:

- Contexto: Es el ámbito donde el patrón de diseño residirá
- Problema: Se debe definir la situación a la que se enfrentan los desarrolladores.
- Motivación: Argumentar las razones que dan pie al problema, la solución y su utilización.
- Solución: Explicación de la solución y detallar la solución en cuanto a sus elementos y estructura

- Estrategias: Explicación acerca de la implementación del patrón de diseño.

- Consecuencias: Los trade-off (ventajas y desventajas) cuando se aplica el patrón.

- Intención: El objetivo específico que persigue.

- Aplicabilidad: Explicar los casos en los que se usa y de que forma.

Gracias a los patrones de diseño los desarrolladores obtendrán necesariamente varios beneficios, por el hecho de que contienen una estructura definida que les permitirá identificar con que tipo de problema están enfrentándose al momento de identificar el patrón que se este utilizando, esto también les permitirá obtener un mayor nivel de abstracción en el diseño, y la reutilización de diseño para aquellos componentes que contengan similitud con el problema

A continuación explicaremos de manera general algunos patrones de diseño que podríamos requerir en el desarrollo de una aplicación Web. Una explicación más detallada vendrá en el capítulo cuatro ya que tengamos un caso específico en el cual podamos identificar el problema y el patrón que ayudara a su solución.

3.9.1 Proxy

Cuando no se puede contar con algún elemento de desarrollo por alguna razón, cualquiera que esta sea, el equipo de desarrollo no se puede detener esperando a que este elemento este listo, para ello existe el patrón conocido como Proxy que emulara el funcionamiento del elemento faltante y permitirá al equipo de desarrollo seguir trabajando sin perder tiempo.

3.9.2 Session Façade

Cuando las operaciones entre el cliente y el servidor son demasiadas, esto aunado a un estilo de programación que involucra directamente los objetos de negocio con las llamadas del cliente se exponen éstos (los objetos de negocio) a ser interceptados por terceras personas para hacer uso malicioso de esta información. Cuando esto ocurre o se prevé es aconsejable hacer caer estos objetos de negocio en un patrón conocido como Session Façade.

Principalmente el funcionamiento del Session Façade se basa en clases que encapsulen la lógica de negocio las cuales oculten al cliente aquellos procedimientos que actúan a partir de los componentes de negocio, permitiendo en consecuencia, la flexibilidad de realizar cambios en la lógica de negocio sin afectar el resultado final ya que se encuentran independizados estableciendo así un mínimo de acoplamiento.

Si se esta desarrollando una aplicación compleja pueden coexistir varios objetos de este tipo y aún cuando se este desarrollando una aplicación más simple, resulta muy útil, aunque aparentemente en su funcionamiento haga las veces de un Proxy más que de un Façade, porque toda aplicación es susceptible de ser mejorada y la introducción de un Façade permitirá realizar cambios posteriormente sin afectar la capa cliente tomando cada vez una forma más aproximada a lo que este patrón de diseño sugiere.

3.9.3 Data Access Object (DAO)

Para lograr la comunicación entre una aplicación y la fuente de datos es necesaria toda una tecnología que permita el acceso a datos. Esta tecnología es constituida por una Interfase de acceso a datos y un API de conectividad con sistemas de bases de datos relacionales.

En este caso existe una gama a partir de la cual podemos hacer una elección de acuerdo a los requerimientos del sistema a implementar, a continuación explicaremos brevemente las alternativas y las razones por las cuales elegimos DAO y no las otras opciones.

RDO (Remote Data Objects) es la interfase diseñada en función del API ODBC. RDO es una tecnología creada por Microsoft para trabajar bajo Visual Basic Enterprise y los resultados que arroja no son mediante RecordSets.

ADO (ActiveX Data Objects) es la interfase de alto nivel de Microsoft que trabaja con otro API de conectividad que es OLE DB y puede ser utilizada por gran variedad de lenguajes de programación incluso Java a través de un elemento común como lo es ODBC, además de que permite trabajar a diversos niveles en la arquitectura cliente – servidor.

DAO (Data Access Object) es una interfase creada para trabajar con la tecnología Microsoft Access sin embargo no es a la que nos referiremos a continuación ya que el API de conectividad a bases de datos de Java **JDBC** es mucho mas flexible y su manera de uso varia de acuerdo al esquema de programación que se utilice, es por ello que bajo el esquema que estamos proponiendo utilizar en este proyecto se hace uso de un patrón de programación igualmente conocido como **DAO** que nada tiene que ver con la de Microsoft, pero que nos va a servir como la interfase programática para interactuar con el **API JDBC** a través de recordsets, fields, properties, etc.

ADO y DAO resultan ser interfases de alto nivel y los beneficios que proveen a un proyecto son considerables, pero la balanza se inclina de manera definitiva hacia DAO debido a que la tecnología ADO al estar dirigida por Microsoft se conlleva a la perdida de **portabilidad** que es motivo de uso de la tecnología JAVA y JDBC al permitir a la aplicación ser útil en diversos sistemas operativos lo cual hace a un sistema **portable,** esto en virtud de que no se ha definido aún la plataforma en que será implementado.

> Los objetos DAO son realmente la mejor forma para el acceso a la base de datos. Ya que ofrecen un modelo simple de programación que puede también proveer una avanzada funcionalidad. Pero no son, sin embrago, los primeros de este tipo. No son tampoco una nueva idea, sino más bien una idea práctica.[9]

Los problemas que impulsan al equipo de desarrollo a hacer uso de este patrón radican en que se hace uso de un elemento externo a la arquitectura Java, nos referimos a las bases de datos y lo que estas conllevan como el lenguaje de consulta (SQL) del que hacen uso, ya que no están estandarizados y aunque son parecidos las diferencias provocan una serie de problemas en el desarrollo y mantenimiento de un sistema. Este problema se agudiza por ejemplo si existe un cambio de proveedor o hasta de versión de la bases de datos.

Como vimos anteriormente el API de JDBC permite acceder de una forma estandarizada a los servidores de bases de datos relacionales, pero, el lenguaje SQL suele tener variantes de acuerdo con el proveedor.

Razones para hacer uso de un DAO sobran, es por ello que es un patrón que se utiliza comúnmente bajo JDBC.

En el caso de las bases de datos no relacionales se agudiza el problema porque no se puede hacer uso del API de JDBC y el código esta mayormente acoplado entre la aplicación y el acceso a la base de datos.

[9] http://www.itver.edu.mx/comunidad/material/tallerbd/apuntes/2.4_Txt.htm
Articulo: Acceso Programático Mediante ADO y JDBC

Para evitar los problemas expuestos anteriormente se hace necesaria la creación de un componente independiente intermedio que logre separar y encapsular la lógica de negocio y la lógica de acceso a la base de datos con el objetivo de reducir el acoplamiento.

Una situación sumamente benéfica de los DAO's es que permiten una configuración mucho más sencilla al momento de hacer el cambio de base de datos ya que en todo caso se realizan algunas modificaciones al DAO y no se afecta el resto de la aplicación.

3.9.4 Transfer Object

Los JavaBeans son utilizados para hacer disponible información al cliente o bien para transportar información del cliente hacia la aplicación, de tal forma que cuando se hace la petición o entrega de un dato la aplicación hace un llamado al método get o set del bean, según sea el caso. Esta es una manera de tener disponible la información para que otra clase haga uso de ella (persistencia). Sin embargo tiene el inconveniente que se debe realizar un llamado para cada dato que se desee extraer y cuando son muchos valores por extraer y muchos clientes que lo solicitan, esta cantidad de llamados afectan el desempeño de la aplicación, a menos que, se implemente el bean mediante el patrón de diseño Transfer Object.

El bean por si sólo nos permite reducir el acoplamiento de una aplicación cuando se articula con el uso de JDBC, DAO's, Session Façade, entre otros objetos y patrones.

La situación problemática ocurre cuando un método con muchos argumentos tiene que ser modificado, ya que las modificaciones no se harán solamente al método sino que se deberán hacer directamente a los llamados a este método esto trae consigo problemas de mantenimiento y acoplamiento. Por otro lado si cada invocación remota a métodos para obtener y brindar información (get y set) se hace para cada argumento correspondiente a cada atributo del objeto reducirá considerablemente el desempeño de la aplicación.

Lo que el patrón de diseño Transfer Object propone es encapsular todos los atributos relacionados, con la idea de que aquellos métodos que los invocan no se hagan de manera separada sino que sólo se realice un llamado al conjunto que contiene la colección de datos que requerimos. Con esto se reduce la necesidad de varias invocaciones remotas a sólo una y permite hacer cambios a los argumentos en el caso de que los atributos son modificados y de esta forma no será necesario hacer modificación alguna a los llamados al método afectado.

La manera en que el Transfer Object funciona es bivalente ya que esta diseñado tanto para contener información entrante (getters), como para concentrar información saliente (setters) y es una fase intermedia entre la información recopilada del cliente y los DAO´s o de manera inversa de la información proveniente de los DAO's que será presentada al cliente.

Capítulo 4

DISEÑANDO UNA PROPUESTA DE SOLUCIÓN PARA LA VENTA Y RESERVACIÓN DE BOLETOS DE AUTOBÚS.

Retomando el ejemplo recurrente en los capítulos anteriores respecto a la venta de boletos de autobús presentamos a continuación el proyecto que da sustento a nuestra tesis.

4.1 Motivación del Proyecto

La empresa nacional de auto-transporte foráneo requiere la modernización de su servicio de comercialización, ya que existe cierto rezago de los métodos del servicio de venta de boletos para viajar en autobús ante las nuevas alternativas de comercio electrónico.

Un elemento esencial que forma parte de la solución a esta problemática es la construcción de un sitio Web el cual provea información al cliente acerca de los servicios que la empresa provee y además, permita hacer uso de algunos de estos a través del sitio, para conseguir con esto un doble beneficio, la difusión del servicio hacia los cibernautas e introducir a la empresa en el mundo del comercio electrónico.

4.1.1 Factores que provocan el desarrollo del Proyecto

- Un servicio telefónico insuficiente.
- Servicio telefónico restringido al uso de tarjeta de crédito.
- Los inconvenientes que trae el comprar el boleto directamente en taquilla.
- El comercio electrónico.

4.1.2 Situación Actual

La empresa nacional de auto-transporte foráneo (ENAF), ha observado que sus competidores han incrementado su nivel de pasajeros en cada uno de sus viajes, mientras que se ha provocado una disminución de estos en dicha empresa desde que la competencia implementó cierto servicio en Internet, el cual ha permitido a los clientes revisar los viajes disponibles y los horarios de las salidas, y en algunos casos hasta la reservación de sus lugares por medio de este servicio.

Actualmente la ENAF ofrece estos mismos servicios vía telefónica, sin embargo, en los últimos meses ha recibido una cantidad importante de quejas derivadas de su servicio telefónico de venta de boletos, debido a que los operadores telefónicos hacen esperar mucho tiempo a los clientes, hasta que estos se desesperan y no concluyen su trámite. Lo anterior proviene de la gran cantidad de clientes que desean hacer uso del servicio, lo cual ha provocado que sea insuficiente la atención proveída por el servicio telefónico para la cantidad de clientes que lo solicitan.

La forma típica de operación en la cual un cliente acude directamente a la taquilla ubicada en la central camionera a partir de la cual iniciará su viaje, por ejemplo, la Central Camionera de Taxqueña en la Ciudad de México con destino a Huatulco en el horario de las 11 de la mañana, donde no existe la certeza para el cliente de que haya la disponibilidad de un viaje (salida) en ese preciso horario, sino que el cliente deberá esperar a la más próxima, esto será, en el mejor de los casos, si es que existe disponibilidad de asientos para este viaje. Aquí se presenta una pérdida de tiempo "inevitable" si consideramos que no hay manera de saber la información precisa de los viajes. Una persona más organizada o con más tiempo suele conseguir su boleto con antelación para llegar en el momento justo de abordar el autobús y evitar la espera, sin embargo en este caso se tendrán que dar dos vueltas, una para comprar el boleto y otra para el momento del viaje, lo cual también representa una situación incómoda.

Existe una alternativa que es la compra vía telefónica, pero en ocasiones resulta complicado porque no podemos apreciar la gama de alternativas a elegir, además de no poder apreciar de manera clara lo que estamos comprando y no queda más que confiar en el operador, quien no siempre tiene la paciencia para solucionar todas las inquietudes que uno como cliente puede tener.

4.1.3 Requerimientos

Para tener claro qué es lo que se necesita, debemos comprender claramente qué es lo que tenemos, para así dirigirnos hacia lo que queremos obtener

4.1.3.1 Lo que actualmente es:

La venta de boletos para el servicio de transporte foráneo se realiza de manera personal acudiendo a una taquilla o a través de un operador telefónico.

Se consulta el origen y el destino del viaje, así como los horarios que les corresponden.

El operador o aquel que atiende en la taquilla, debe indicar qué lugares se encuentran disponibles para que el cliente elija.

Finalmente el cliente procede a realizar su pago si es en taquilla, o bien provee los datos necesarios para hacerle el cargo a su tarjeta, en caso de que se esté utilizando en servicio telefónico.

Como ya se mencionó previamente, ambos modos presentan inconvenientes, por un lado el adquirir un boleto directamente en taquilla no garantiza que podrá hacerse el próximo viaje, sino el que esté disponible.

Si se adquiere vía telefónica se requiere forzosamente de tarjeta de crédito y se le confía información valiosa a un tercero, además de que no se puede visualizar lo que uno está comprando, en lo que se refiere por ejemplo a la ubicación del asiento que se nos asigne.

Si bien es cierto que así es como se han hecho las cosas hasta el momento y los resultados han sido satisfactorios para la empresa, no quiere decir que no puedan hacerse de una manera mejor.

4.1.3.2 Lo que debe ser

Se intenta introducir un servicio que ofrezca a los clientes un servicio más cómodo, el cual suprima los inconvenientes que las otras alternativas traen consigo, haciendo uso de los más recientes avances tecnológicos orientados al comercio electrónico.

El propósito de la introducción de una nueva tecnología no es eliminar las alternativas tradicionales, sino que ésta sea una más de ellas, donde puedan convivir las tres situaciones y así hacer llegar el servicio a una gama más amplia de clientes, donde las necesidades particulares del cliente sean consideradas para hacerle la vida más simple y de paso aligerar la carga de trabajo que las opciones tradicionales pudieran tener en determinados momentos.

Una vez que la nueva tecnología esté en funcionamiento y debido a que no va a contraponerse a la venta de boletos en taquilla o por teléfono, no afectará inmediatamente a estas técnicas, sin embargo, de manera gradual ira formando parte de las preferencias del cliente, pues podrá adquirir sus boletos por este medio en lugar de los otros, lo cual será más cómodo. Por otro lado los clientes asiduos a otras empresas, al notar estas ventajas podrían hacer uso de estos servicios y por tanto se incrementará en cierta forma la venta del servicio proporcionado. Será entonces una especie de escaparate en donde la empresa podrá mostrar la calidad de sus servicios y de esta forma captar un mayor número de clientes.

4.1.4 El elemento renovador

En una empresa con un funcionamiento de mercadeo tradicional, el cual cuenta con una competencia amplia, que ofrece el mismo tipo de servicio y con precios muy similares, de hecho poseen cierto equilibrio entre si, pero si existe un elemento extra que le sea atractivo al cliente necesariamente romperá este equilibrio.

La magnitud que representa la adición de una tecnología de este tipo en el mercado no tiende a ser algo trivial, es un elemento que sin duda lleva a la empresa a un ámbito renovado porque se está entrando en la forma en la que el comercio se lleva a cabo en nuestros días y que tiende a ser la forma en que se harán las cosas en el futuro, es decir, que se estará entrando en el ámbito del comercio electrónico, el cual constituye una de las fases más importantes en el mundo de los negocios electrónicos y que le dará una proyección y una amplitud favorable a la posición de la empresa en el mercado, en comparación con su sistema de comercio tradicional.

4.2 Estrategia de Solución

El problema a solucionar radica esencialmente en el hecho de que un negocio tradicional al competir contra otros negocios, que adicionalmente han entrado a los negocios electrónicos, estará en desventaja.

El tema que involucra directamente al cliente con la empresa podemos indicar que es la parte del e-business que está involucrada a este rubro es el e-commerce en su modalidad business to consumer o b2c.

Uno de los primeros pasos para reformar la estructura de una empresa es el comercio electrónico porque va a permitirle llegar a una mayor cantidad de clientes en potencia.

La empresa necesita crear una forma auxiliar para venta y reservación de boletos que sea una alternativa tal que reduzca la afluencia al servicio telefónico, y la cual represente una opción para que cualquier persona tenga acceso a ella de manera sencilla, desde cualquier parte, de ser posible desde su casa y mediante la cual sea posible consultar los viajes programados para el destino deseado, además de que ofrezca la posibilidad de reservar uno o más lugares para el viaje seleccionado e incluso comprar los boletos.

Ofrecer una manera sencilla podría decirse que lo que se necesita es ofrecerles a los clientes una forma menos complicada de obtener sus boletos o al menos asegurar su lugar en el viaje que desee. Para lograr este objetivo la empresa requiere el uso de las tecnologías más actuales, debido a que la infraestructura tecnológica de la empresa está en ese mismo rumbo y la intención es que convivan ambas, para que en un futuro ofrezcan los mayores beneficios y calidad en el servicio al cliente, al ser éste sencillo, rápido y confiable.

4.2.1 Requerimientos de la solución al problema

A partir del problema podemos obtener de manera concreta que se requiere de una alternativa que tenga las siguientes características:

- Haga uso de la tecnología más actual
- Permita acceso a los clientes de manera directa e irrestricta

- Simplicidad y sencillez de operación
- Servicio Rápido y Confiable
- Consulta de viajes que la empresa ofrece.
- Reservar o apartar el lugar o lugares que el cliente desee ocupar.
- Comprar los boletos que el cliente solicite
- Evitar al máximo la intervención de terceros (empleados) durante el trámite.

4.3 Propuesta de Solución

La solución que se propone a continuación contará con las consideraciones hechas por la empresa en los tópicos anteriores, y como se podrá apreciar, la tendencia está claramente apoyada en lo que propone el comercio electrónico en su modalidad de negocios a consumidor (b2c).

4.3.1 Panorama General de la Solución

¿Qué se va a hacer?

Diseñar y Construir un sitio en Internet que permita hacer uso de los servicios de Consulta de Viajes, Reservación y Compra de boletos de Autobús, dirigido a clientes que acostumbren el uso de sitios de compra en línea.

¿Mediante qué?

Mediante la creación de una página Web que estará construida bajo la arquitectura Java, la cual brindará la potencialidad y flexibilidad que ofrece esta arquitectura.

¿Para qué?

El motivo esencial es la creación de una herramienta que contribuya a introducir a la empresa en el mundo del comercio electrónico, permitiendo al cliente realizar las operaciones básicas con las que la empresa efectúa la comercialización de su servicio.

4.4 Funcionalidad de la solución

4.4.1 Objetivo del Proyecto

Proporcionar un sitio Web de venta y reservaciones sustentado en la plataforma Java, la cual le proveerá la potencia y flexibilidad necesaria para atender los 365 días del año, las 24 horas del día, de manera segura y eficiente a aquellos clientes que deseen hacer uso del servicio vía Internet.

4.4.2 Descripción de los elementos que intervendrán en el funcionamiento

El sitio Web deberá ser accesible vía Internet y permitirá el acceso a todos los clientes del negocio por este medio, quienes podrán tener acceso a los siguientes servicios:

Consulta de Viajes:

Se le llama viaje al recorrido que cubrirá un autobús en determinada ruta, fecha y horario. Por tanto la consulta de viajes estará determinada por estos tres elementos. Este servicio lo provee el **sitio.**

Registro de Cliente:

Aquel **Cliente** que desee hacer uso del sistema de reservación o venta de boletos deberá registrarse proporcionando su nombre y dirección de correo electrónico al **sitio**.

Selección de Asientos:

El **sitio** debe ser capaz de marcar aquellos asientos que sean reservados como ocupados (bloqueados) y comunicar este cambio a la **empresa** o en el caso de la venta, que estén en espera de la autorización del **banco**.

Desbloqueo de Asientos:

Este evento ocurrirá cuando el asiento reservado esté de nuevo en disponibilidad (desbloqueado) cuando no se cumplan las disposiciones establecidas por la **empresa** o en el caso de que la autorización del **banco** no se lleve a cabo.

Reservación en Línea:

El **cliente** podrá elegir los asientos que desee comprar en taquilla, mediante el servicio de reservación, el cual bloqueará los asientos y de esta forma quedarán a disposición de quien reserva, siempre y cuando se cumpla con las restricciones establecidas por la **empresa**.

Venta en Línea:

El **cliente** podrá comprar los boletos correspondientes a los asientos que haya elegido mediante el uso de su tarjeta de crédito, donde el **sitio** se comunicará con el **Banco** para solicitar el pago, el **banco** deberá entonces aceptar o rechazar la transacción y notificar el resultado al **cliente** por medio del **sitio**.

Conexión banco:

El **Sitio** se comunicará con este servicio para establecer si una transacción es aceptada o no.

Envío de Correo:

Cuando una reservación o venta sean concluidas, el **sitio** deberá enviar un correo al **cliente** donde se incluirán los detalles de la transacción.

Interfase de Conexión:

El **sitio** debe tener acceso al sistema que gestiona los viajes y demás servicios (**empresa**) estableciendo una conexión que se realizará por medio de esta interfase.

Pago con Tarjeta de Crédito

En este punto el **Cliente** introducirá los datos que el **sitio** le pida para efectuar una transacción bancaria mediante el uso de su tarjeta de Crédito

Exclusiones:

El **sitio** se debe regir de acuerdo a las políticas propias de la **empresa,** en cuanto a cantidad de boletos que se vendan o reserven mediante esta modalidad, y los descuentos disponibles para cada tipo de **cliente**.

4.5 Análisis del Sistema de Venta y Reservación de Boletos de Autobús.

El sistema que nos proponemos implementar está constituido por un sitio en Internet al cual se podrá acceder por medio de una página Web, el cual permitirá realizar consulta de viajes, compra y reservación de boletos en línea vía Internet a sus clientes.

Los actores que intervienen en este sitio son 4:

- Cliente – Se refiere a aquella entidad que accede al sitio mediante la página Web y hace uso de los servicios que ya hemos mencionado.
- Sitio[1] - Es la entidad informática encargada de gestionar los servicios a los que nos referimos.
- Banco – Es la entidad informática de proveer los servicios del banco.
- Empresa – Esta entidad informática se encarga de proporcionar la información correspondiente a los viajes, disponibilidad, bloqueo y desbloqueo de asientos y reservaciones.

[1] Es posible que se malinterprete el hecho de que pongamos al sitio, banco y empresa como actores, ya que estamos habituados a ver a un actor como un ente físico que rápidamente identificamos como una persona, sin embargo es correcto el situar como un actor a una entidad no necesariamente personificada.

4.5.1 Casos de Uso

El análisis estará sustentado en los casos de uso que serán hechos a manera de narrativa y a partir de los cuales se derivarán ciertos diagramas que darán paso al diseño de la solución.

Por principio de cuentas el trabajo de análisis y diseño arrojan como resultado diagramas entidad-relación y de clases que junto con el prototipo serán los elementos a partir de los cuales podremos dar inicio al desarrollo y construcción del sitio.

Las Narrativas y diagramas que aquí mencionamos podrán ser ubicados en el Apéndice, donde estarán aquellos elementos concernientes al análisis del Sistema propuesto.

Casos de Uso – Cliente

- Consulta de Viajes
- Registro de Cliente
- Disponibilidad de asientos
- Selección de Asientos
- Reservación en Línea
- Venta en Línea

Casos de Uso – Empresa

- Consulta de Viajes
- Reservación en Línea
- Venta en Línea
- Selección de Asientos
- Desbloqueo de Asientos
- Disponibilidad de asientos
- Interfase de Conexión

Casos de Uso – Banco

- Interfase de Conexión
- Conexión Banco
- Pago a Tarjeta de Crédito
- Venta en Línea

Casos de Uso – Sitio

- Consulta de Viajes
- Registro de Cliente
- Reservación en Línea
- Venta en Línea
- Disponibilidad de asientos
- Selección de Asientos
- Desbloqueo de Asientos
- Conexión Banco
- Envío de Correo
- Interfase de Conexión
- Pago a Tarjeta de Crédito

4.6 Arquitectura Tecnológica de la Solución

La aplicación Web que nos proponemos realizar deberá seguir los principios de la arquitectura cliente-servidor, en su modalidad de multicapas, debido a que es una de las mejores alternativas para sustentar una aplicación orientada a la Web, y debe cumplir con los requisitos de la modalidad de comercio a consumidor de los negocios electrónicos, que al ser dos entidades comunicadas vía Internet requieren de una respuesta casi inmediata o que se procese de la manera más veloz posible, ya que la respuesta deberá viajar por la red para ser presentada de nuevo al cliente y por tanto al trabajar bajo este esquema se reparte el procesamiento entre los diferentes equipos o capas.

Primera Capa

Esta capa estará constituida por los elementos de presentación y lógica de negocio que si bien es posible dividirlos en 2 capas, para los fines de nuestra aplicación no será necesario, ya que el contenedor Web administra ambos elementos sin que se vea afectado el rendimiento de la aplicación, ya que su diseño así lo contempla.

Cuando hablamos de los elementos de presentación nos referimos de manera concreta a las páginas creadas con HTML, JavaScript, Imágenes, y algunos otros elementos de diseño gráfico como Flash, etc.

Al referirnos a la capa de lógica de negocio nos referimos a aquellos elementos que darán al sitio funcionalidad, estos elementos son en nuestro caso concreto archivos tipo Java Server Pages, *Servlets*, XML, y clases de JAVA.

Las siguientes capas corresponderán a los sistemas de información.

Segunda Capa

En esta capa residirá la *Base de Datos* a partir de la cual el sitio logre desplegar e introducir en parte la información que el sitio requiera. En este punto será necesaria la utilización de un servidor de archivos como *apache*, el cual permitirá acceder desde otro equipo a la *Base de Datos*.

Tercera Capa

En la presente capa se encuentra el sistema de venta y reservación en Línea que da vida y funcionalidad a la empresa, con el cual se deberá comunicar el sitio para obtener toda la información que será desplegada al cliente como consulta de viajes, disponibilidad de asientos, etc.

Cuarta Capa

Una capa más es la concerniente al sistema que proporciona los servicios del banco, los cuales serán solicitados también por el sitio para ejecutar las transacciones de Venta en Línea a partir de la solicitud de crédito para el cliente.

4.7 Diseño de la Solución

4.7.1 Arquitectura de Software para la solución.

En la primera capa, la parte de la presentación, resulta ser una tarea simple, desde el punto de vista funcional, sin embargo, es una parte que requiere ser abordada con mayor detalle por el equipo de diseño gráfico, que es el más apto para esta tarea. De tal suerte que si se cuenta con dicha ayuda, será posible iniciar el trabajo de los desarrolladores de la aplicación, que en base a las páginas Web que entregue el equipo de diseño, podrá establecerse lo que conocemos como prototipo, a partir del cual se podrá visualizar el funcionamiento "virtual" de la aplicación, es decir, que simulará la manera en que trabaja; aunque en realidad no esta generando proceso alguno.

La otra parte que complementa la primera capa es la parte de la lógica de negocio, en la cual se encuentra la parte "pesada" del trabajo de los desarrolladores, porque es aquí donde reside toda la funcionalidad del sitio mediante las Java Server Pages y las demás clases Java que estarán encargadas del acceso a *Base de Datos*, la intercomunicación con el servicio de reservación y venta de boletos, además de la comunicación con el banco.

Lo anterior es necesario explicarlo para visualizar donde está el núcleo de lo que nos proponemos realizar. La arquitectura de software tiene sustento práctico en esas Java Server Pages y clases Java que representan el interés de nuestro trabajo como desarrolladores y el motivo esencial de esta investigación.

4.7.2 Prototipo

Lo que el prototipo nos indica es que le será presentada al cliente una página HTML, la cual le permitirá al cliente elegir un origen, un destino y la fecha en que desea realizar su viaje, una vez que el cliente hace la elección se entregará esta información a un *Servlet* que será quien atrape la información, de ahí se entregará la información a un *Validador* que verificará la longitud y estructura de la información, se llamará un objeto tipo transfer, para almacenar la información en su estructura, esta información se dirigirá hacia un *Proxy* para que éste almacene la información y la redireccione hacia el componente de interfase de comunicación, la cual se encargará de crear la trama que será entregada al servicio de la empresa encargado de entregar los viajes correspondientes a los datos recibidos

4.7.3 Propuesta de Operación del Sitio

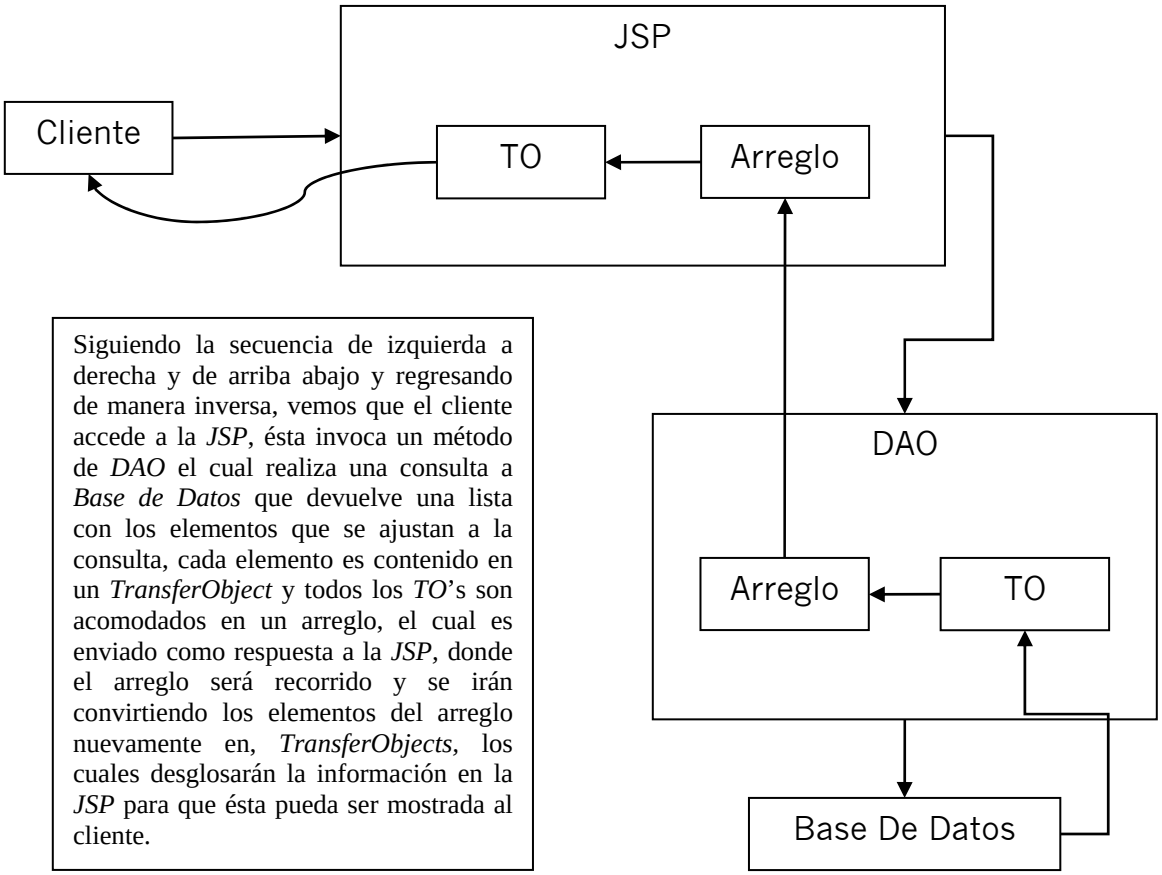
Veamos ahora como será el funcionamiento del sitio de acuerdo a nuestra propuesta.

4.7.3.1 Accediendo al Sitio Vía Internet

En primer término el cliente requiere interactuar con el servicio que ofrecemos, por tanto este iniciará su explorador de Internet, tecleará en la barra de direcciones la correspondiente a nuestro sitio, por ejemplo: www.enaf.com.

Una vez que el cliente acceda al sitio se estará ejecutando una *JSP,* la cual presentará un listado de orígenes y destinos tomados en ese preciso instante de la *Base de Datos* a través de un objeto tipo *DAO*, éste almacenará la información en un arreglo[2] de Objetos tipo Transfer (Transfer Object) y este le devolverá la información recopilada a la *JSP* presentando de esta forma la página principal <home> del sitio.

Diagrama Accediendo al Sitio Vía Internet



Siguiendo la secuencia de izquierda a derecha y de arriba abajo y regresando de manera inversa, vemos que el cliente accede a la *JSP*, ésta invoca un método de *DAO* el cual realiza una consulta a *Base de Datos* que devuelve una lista con los elementos que se ajustan a la consulta, cada elemento es contenido en un *TransferObject* y todos los *TO*'s son acomodados en un arreglo, el cual es enviado como respuesta a la *JSP*, donde el arreglo será recorrido y se irán convirtiendo los elementos del arreglo nuevamente en, *TransferObjects,* los cuales desglosarán la información en la *JSP* para que ésta pueda ser mostrada al cliente.

[2] Arreglo es una estructura de programación utilizada para contener información de manera secuenciada en una, dos o n dimensiones. Más adelante haremos uso de la expresión ***recorrer un arreglo***, esto es debido a que la información contenida en un arreglo es susceptible de ser recuperada mediante una estructura de control cíclica encargada de explorar el contenido del arreglo y así disponer de la información que se obtenga.

A partir de la página principal el cliente podrá empezar a interactuar con el sitio, primeramente el sitio le pide al cliente elegir el origen, destino y fecha en que desea realizar el viaje, una vez que ha realizado su elección, el cliente pulsará el botón de consulta de viajes.

4.7.2.2.2 Proceso de Consulta de Viajes

- La información que la *JSP* ha recopilado será atrapada por un *Servlet* y éste se encargará de dirigir el flujo de la información.

- El *Servlet* direccionará estos datos hacia una clase que llamaremos *Validador,* el cual se encargará de revisar que la información que fluye a través del sistema sea correcta y creará un objeto de tipo transfer para contener los datos validados, estos son atrapados nuevamente por el *Servlet*.

- El *Servlet* creará un objeto tipo *Proxy* e invocará uno de sus métodos, enviándole como parámetro el objeto *Transfer*.

- El método del *Proxy* estará encargado de crear un objeto tipo *DAO.*

- El DAO se encargará de formar una expresión (sentence), que con la ayuda de otro objeto que es un componente perteneciente a la interfase de comunicación direccionará dicha expresión hacia un *Socket.*

- El Socket se encargará de formar la trama y hacer la petición de la información (consulta), con la que se podrá ejecutar una consulta al servicio de la empresa.

- La empresa devolverá por este mismo medio (*Socket*) una respuesta, en donde el resultado será manipulado para ser almacenado en un arreglo, el cual estará integrado como una colección de resultados (resultset).

- El resultset será entregado al *DAO,* que además se encargará de desmenuzar la información en un arreglo de viajes, que se enviarán al *Proxy.*

- En el Proxy se organizará esta información en un arreglo de objetos tipo transfer en donde cada objeto contendrá la información correspondiente a un viaje, para así entregarle al *Servlet* un arreglo de *TransferObjects.*

- El *arreglo* será incrustado en el recurso de petición (request) del *Servlet* y de esta forma direccionará hacia la *JSP* para poner a su disposición los elementos del arreglo mediante el recurso de respuesta (response) del *Servlet*.

- Con la información contenida en la *JSP* se podrá ejecutar un ciclo de instrucciones que desplieguen la información en la página y de esta forma el usuario pueda visualizar el resultado y seguir interactuando con el sitio.

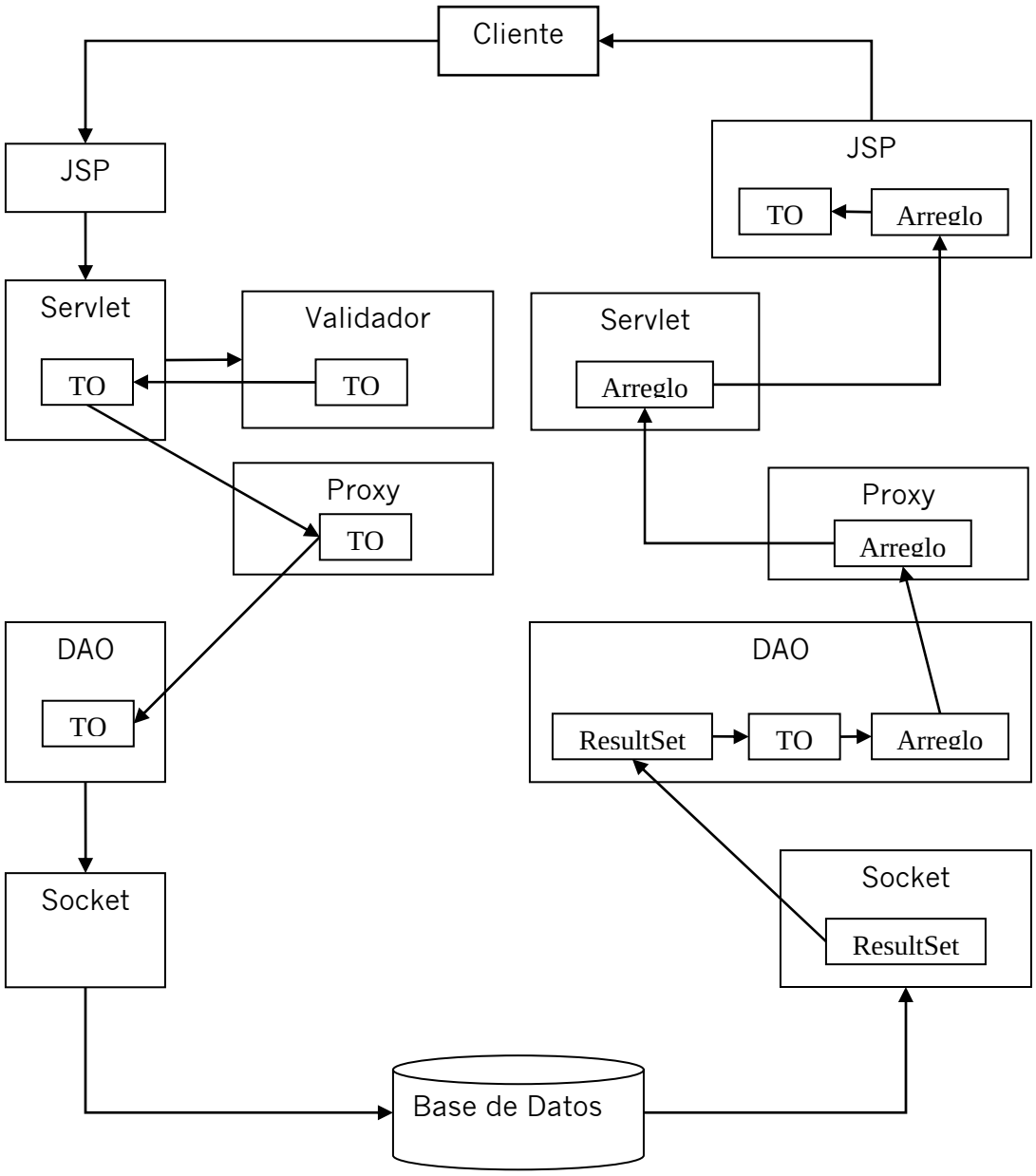
Después de toda esta explicación tan técnica, si no es entendida, no se debe estar aterrado por la consecución de elementos poco familiares a los que nos encontramos y todos mezclados de forma tan abrupta, ya que esto lo iremos explicando paso a paso.

Este procedimiento se repetirá para el resto de los servicios que provee la empresa como pueden ser reservación, venta, etc.

En los últimos párrafos se hizo todo un comprimido, con todos aquellos elementos que se han estado explicando a lo largo de la investigación y aún cuando es posible que estos elementos por separado no nos permitan concebir su operación conjunta, tenemos el compromiso de ejemplificar todo este proceso, el cual nos proponemos desmenuzar, ya en un sentido más técnico, enfocado a la programación usando el lenguaje Java. Sin embargo no era posible entrar a estos tópicos si todo lo anterior no quedaba establecido como referencia.

El siguiente diagrama permitirá visualizar gráficamente como es que ocurre el flujo de la solución que proponemos.

Diagrama Consulta de Corridas



4.8 HTML

El lenguaje HTML como ya se ha definido permite presentar una página Web a través de una Red como Internet mediante el protocolo de transferencia HTTP el cual es interpretado por una aplicación conocida como Explorador Web o Web Browser.

Típicamente la construcción de la apariencia de una página Web esta delegada a un equipo de Diseño Gráfico, sin embargo, debemos tener claro como funcionan los componentes que forman parte de la página ya que a partir de ellos se llevara a cabo un proceso de retroalimentación con la capa de lógica de negocio que implementaremos.

4.8.1 Sintaxis

La estructura básica de una página HTML es la siguiente:

```
<HTML>
<HEAD>
<TITLE> Titulo de la Pagina </TITLE>
<META NAME="Generator" CONTENT="">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
</BODY>
</HTML>
```

Apreciamos aquí que los elementos están encerrados entre los signos de menor que "<" y mayor que ">" y esto forma parte de la sintaxis del lenguaje HTML, a estos elementos encerrados entre los signos se les conoce como etiquetas o tags que hacen referencia a la M (Mark-up) en las siglas de HTML, por ello decimos que es un lenguaje de marcas.

Cada una de esas etiquetas puede contener atributos, es decir, características que definen su apariencia y comportamiento. En el ejemplo los identificaremos como NAME= que establecerá el nombre, aunque podemos encontrar en este mismo sentido atributos como TYPE para el tipo, VALUE para el valor, COLOR, etc.

Es posible encontrar etiquetas de apertura y cierre como las que definen el título de la página del ejemplo (<TITLE> Titulo de la Pagina </TITLE>), éste es definido entre las dos etiquetas y la etiqueta de cierre se identifica por la diagonal.

Por último definiremos un control que nos permitirá obtener información de la página. La apariencia del control esta definido por el atributo TYPE mientas que la etiqueta para definir un control donde se introducirán datos será <INPUT>.

Ejemplos:

```
<INPUT TYPE="text" NAME="">
<INPUT TYPE="password" NAME="">
<INPUT TYPE="button" NAME="">
```

4.9 JAVA

4.9.1 Java Server Pages

Una página HTML es una interfase gráfica de usuario y para ello es recomendable asignar esta tarea a los diseñadores gráficos que le darán a la página una apariencia atractiva y de fácil utilización para ellos (los usuarios) y que en determinados casos será aquello que integrará el *prototipo* de nuestra aplicación.

A partir de este prototipo nos será posible construir una JSP ya que este tipo de tecnología esta basada en HTML, de manera más precisa podemos decir que una página HTML es susceptible de ser modificada agregando código JAVA para así proporcionarle la funcionalidad necesaria con la finalidad de acceder a fuentes de datos para suministrarle un comportamiento más complejo (dinámico). Cuando incrustamos código Java a una página HTML estamos convirtiéndola en una página JSP y por tanto la extensión del archivo .HTML será sustituido por .JSP.

Es posible importar librerías, instanciar objetos, ejecutar métodos, declarar variables, prácticamente de la misma manera que en una clase.

Cuando incrustamos código Java en una página HTML se establecen unas etiquetas de este tipo <% para abrir y %> para cerrar, de esta forma el contenedor Web identificará que lo que esta entre los separadores es código Java, lo compilará y ejecutara creando una especie de Servlet virtual.

Cuando utilizamos las etiquetas "<% %>" se dice que estamos introduciendo ***Lenguaje Script*** conforme la especificación de JSP. El lenguaje Script al que nos referimos presenta 4 formas diferentes de incrustar el código de acuerdo a la forma e intención de la instrucción.

1. ***Comentarios*** - <%-- Texto Informativo --%> - Que es únicamente texto que orienta al Desarrollador acerca de lo que hace o debe hacer alguna instrucción.

2. ***Declaraciones*** - <%! Tipo NombreVariable = Valor %> - Se hace uso de esta forma cuando se requiera declarar variables o métodos.

3. ***Expresiones*** - <%= Variable %> - Es un elemento que calcula ciertas operaciones o variables y las convierte en objetos tipo cadena para presentar los resultados de la evaluación como texto en la JSP.

4. ***Scriplet*** – <% Código Java Puro %> - Ocurre cuando deseamos incrustar código Java Puro de una manera idéntica al que se usa en los Servlets y las Clases.

Aparte de los Scripts que acabamos de señalar existen otros elementos como las Directivas, Acciones y Objetos Implícitos.

Las ***Directivas*** son de un carácter más general y proporcionan información y comportamiento global a la JSP.

Ejemplo: `<%@include file="nombreArchivo.txt"%>`

Las ***Acciones*** son elementos que proveen a la página JSP información y comportamiento al momento que esta está siendo procesada para ser mostrada. Tienen además la cualidad de ser elementos personalizables ya que se pueden crear acciones de acuerdo a lo que los desarrolladores requieran.

Ejemplo: `jsp:incluye page=”nombreArchivo.html”`

Finalmente existen ***Objetos Implícitos*** que están disponibles para el desarrollador y puede hacer uso de ellos en cualquier momento sin necesidad de declararlos ya que el contenedor Web los genera y se solicitan directamente desde la JSP. Estos son algunos de los objetos implícitos:

`out, request, response, sesion, page, exception, config.`

4.9.2 Servlet

Los Servlets los utilizaremos para recabar la información proveniente de la página JSP y también para entregarle un resultado a esta misma o bien para generar una JSP nueva.

Cada control en una página HTML o JSP contiene esencialmente un valor y un nombre que lo identifica y lo hace único. El valor puede ser establecido desde que se crea la página o bien puede ser modificado por el cliente, es decir cuando la página es presentada al cliente permite interactuar a él mediante los controles que éste manipule y por ejemplo si en una casilla de texto el cliente introduce información la misma será transmitida a través del atributo VALUE de cada control hacia el Servlet.

Existe un par de elementos comunes entre una JSP y un Servlet se les conoce como httpServletRequest y httpServletResponse que son elementos que permiten la comunicación entre JSP y Servlet ya que el primero es un servicio de petición y el segundo es de respuesta, de tal suerte que la información viaja a través de estos elementos.

HttpServletRequest es un elemento de petición de información del Servlet a la JSP, así que la información que recabe la JSP será entregada por este medio al Servlet.

De manera inversa HttpServletResponse en la manera en que un Servlet entrega información a una JSP.

Un Servlet es también una Clase, pero su comportamiento es dictado por el objeto Servlet (es un objeto definido en el API de Java) del cual extiende su funcionalidad y además requiere de un archivo de configuración que le permita convivir en un entorno Web, este archivo de configuración será codificado en XML y llevará por nombre web.xml y estará posicionado en la carpeta correspondiente a nuestra aplicación dentro del fichero de aplicaciones del contenedor Web.

En una aplicación Web como ésta no existe un método principal (main) a través del cual se desencadene el comportamiento de la aplicación, sin embargo, un comportamiento cercano al método principal es el que opera el Servlet porque constituye la parte medular de la aplicación ya que a partir de aquí se mandan llamar métodos de otras clases se ejecutan y los resultados regresan al Servlet para seguir la secuencia de operación que éste establece para que finalmente el Servlet devuelva el resultado mediante el servicio de repuesta HttpServletResponse llamando a la JSP encargada de presentar los resultados al cliente.

EjemploServlet.java

```
import javax.servlet.ServletException;
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.print.*;

public class EjemploServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
    {
       /*De esta manera se extraen los datos de una JSP
        *Las palabras entre comillas corresponden al
        *atributo NAME de los controles HTML*/

       String nombre=request.getParameter("nombre");
       String apellido=request.getParameter("apellido");
       String edad=request.getParameter("edad");
       String telefono=request.getParameter("telefono");

       /*Para devolver la información a una JSP*/

       request.setAttribute("dato1", nombre);
       request.setAttribute("dato2", apellido);
       request.setAttribute("dato3", edad);
       request.setAttribute("dato4", telefono);
       response.sendRedirect(/confirma.jsp);

       /*La JSP obtiene el nombre invocando el método
        *inverso: request.getAttribute(“dato1”) */
    }
}
```

Aquí presentamos un pequeño Servlet que obtiene los datos de una JSP que tiene 4 controles (nombre, apellido, edad y teléfono) y a través del método doGet extraemos esta información contenida en el objeto request que contiene determinada información proveniente de la JSP a la cual es posible acceder mediante el método getParameter el cual extrae el valor contenido en el atributo VALUE del control al que se hace referencia posicionando entre los paréntesis el valor correspondiente al atributo NAME del control de la JSP. Finalmente el dato entregado por getParameter lo almacenamos en una variable tipo cadena (String).

En la Segunda sección del método doGet() nos proponemos a reenviar esta información a otra JSP usando el método setAtribute(“dato1”, Nombre) dónde el primer elemento dentro de los paréntesis es donde será contenida la información y el segundo es la variable que contiene la información dentro de la clase, de tal suerte que cuando la JSP a la que se está invocando con el método send.redirect() solicite la información deberá llamar el método getAtribute(“dato1”) donde dato1 contendrá la información correspondiente a la variable Name en el Servlet.

Como ya habíamos comentado el Servlet tiene un archivo de configuración que reside en el contenedor Web, a continuación editaremos este archivo para dar de alta el Servlet que acabamos de crear y de esta forma configurarlo en el entorno de nuestra aplicación.

WEB.XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>NombreServlet</servlet-name>
    <servlet-class>EjemploServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>NombreServlet</servlet-name>
    <url-pattern>/Ejemplo</url-pattern >
  </servlet-mapping>
</web-app>
```

Como podemos apreciar son dos secciones, la primera ubicada entre las etiquetas <servlet> consta de 2 líneas donde se define un nombre con el que identificaremos al Servlet en la primera línea y el nombre de la clase Servlet que construimos en la segunda.

La siguiente sección corresponde a las líneas que están entre las etiquetas <servlet-mapping> donde la primera línea debe ser idéntica a la primera línea de la sección anterior ya que con esto se identifica al Servlet al que nos referimos para que en la segunda línea finalmente se indique una ubicación y nombre mediante el cual se podrá ejecutar el servlet cuando éste sea invocado por la JSP a través de ese nombre.

Todo servlet deberá ser dado de alta en este mismo archivo WEB.XML de la misma forma que se ejemplificó, las primeras secciones juntas en un bloque superior y las segundas secciones en un segundo bloque.

Una vez hecho lo anterior podemos ejecutar nuestro servlet desde la JSP para ir logrando darle la funcionalidad que requerimos a nuestra página o sitio.

4.9.3 Bean

Una aplicación como la que planeamos desarrollar se basa en la interacción recurrente de clientes con el servicio que se proporciona, donde el cliente obtiene cierta cantidad de información extraída de las fuentes de datos mediante métodos únicos correspondientes al elemento solicitado. De la misma manera los datos que el cliente provee son enviados para que el sistema los procese y de ser necesario los almacene.

El manejo de ésta información es posible gracias a el objeto conocido como BEAN que actúa como un medio de transferencia de la información haciendo uso de métodos de obtención (get()) y métodos de entrega (set()) para cada uno de los atributos utilizados cuyos valores fluyen a través de éste. Sin embargo la invocación a cada uno de estos métodos implica una llamada remota, y si consideramos que son varios datos los que se van a obtener/entregar y que el número de clientes que hacen uso de estos servicios puede ser simultáneo e ilimitado, nos enfrentaremos en algún momento a un gran número de llamados remotos que afectaran irremediablemente el rendimiento de la aplicación creando una sobrecarga en la red que puede originar pérdida de información, lenta circulación de la información e incluso la caída del sistema y esto por razones evidentes significa que el manejo de la infamación en esta forma resulta ineficiente.

Lo anterior nos ofrece ventajas en la manera de transferir la información pero desventajas en el rendimiento de la aplicación en los casos descritos anteriormente.

La manera en que podemos explotar las ventajas de este componente y reducir las desventajas es haciéndolo caer en el esquema planteado por el patrón de diseño conocido como TransferObject.

De manera general un TransferObject (TO) ofrece una manera de encapsular los atributos con el fin de que se acceda a ellos mediante una sola llamada al método de obtención o entrega del TransferObject.

En el momento en que el cliente pide información se crea una instancia que se encarga de construir un TO le introduce los valores correspondientes a los atributos solicitados y devuelve el conjunto de valores de una sola vez, lo cual implica un sólo método a llamar y una única invocación remota al servidor.

Los métodos get() y set() de que hablamos con anterioridad siguen siendo utilizados en el TO, la diferencia radica en que su invocación es de manera local y no de manera remota como se planteaba al principio.

DatosTO.java

```
public class DatosTO {
    /*Declaracion de variables privadas*/
    private String nombre;
    private String apellido;

    /*Definicion del Transfer Object*/
    public DatosTO (String pNombre, String pApellido) {
       nombre = pNombre;
       apellido = pApellido;
    }

    public String getNombre() {
       return nombre;
    }

    public String getApellido() {
       return apellido;
    }

    public void setNombre(String s) {
       nombre = s;
    }

    public void setApellido(String s) {
       apellido = s;
    }

}
```

Como podemos apreciar en la definición de la clase DatosTO primero se declaran las variables que representarán a los atributos con los que se relacionarán, posteriormente se definen tres tipos diferentes de métodos: el primero es la definición del método general del Transfer Object que se encargará de invocar de manera local los otros 2 métodos según corresponda. A los métodos usados para obtener información se les conoce como getters y a aquellos que se utilizan para entregar datos son llamados setters.

La función de los getters es devolver el valor del atributo correspondiente al método que se invoca, dicho valor ha sido introducido previamente por un método setter en el TransferObject.

Los setters al ser invocados reciben como parámetro el valor de un atributo e introducen el valor recibido en una variable que estará disponible cuando se invoque el método getter del TransferObject.

4.9.4 Conexión

Todo DAO requiere hacer referencia a una clase de conexión que le permitirá establecer una comunicación con la base de datos. Por tanto crearemos primeramente esta clase.

ConexionBD.java

```
package conexion;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexionBD {
    public Connection getConexion()throws Exception {
       Class.forName("oracle.jdbc.driver.OracleDriver");
       return
       DriverManager.getConnection("jdbc:oracle:thin:@
                           //stationwork:1521/database",
                                   "system",
                                   "system");
    }

    public void cierraConexion (Connection pCon) {
       try {
          if (!pCon.isClosed()) {
             pCon.close();
          }
       } catch (SQLException e) {
          e.printStackTrace();
       }
    }
}
```

El primer método realiza la conexión, primero indica el controlador (driver) que debe utilizar (OracleDriver) y devuelve un objeto de tipo Connection que es formado por el nombre del equipo en la red (stationwork) de la computadora que contiene la base de datos, el puerto (1521) de acceso, el sid (database), el nombre de usuario (system)y la contraseña (system).

El segundo método cierra la conexión, lo cual es indispensable realizar cuando no se requiere información a partir de ella por razones de seguridad.

LocalidadDAO.java

```
public class LocalidadDAO {

    public ArrayList buscaLocalidad() throws Exception {
       ArrayList arregloLocalidad = new ArrayList();
       Connection conexion = null;
       Statement stmt = null;
       ResultSet resultado  = null;

       String sql = “SELECT loc_clave,” +
                    “ loc_nombre” +
                    “ FROM localidad”;

       try {
          conexion = ConexionDB.getInstance().getConexion();
          stmt = conexionLocalidad.createStatement();
          resultado = stmt.executeQuery(sql);
          while (resultado.next()) {
             arregloLocalidad.add(new LocalidadTO(
                rs.getString("loc_clave"),
                rs.getString("loc_nombre")
             ));
          }
       } catch (Exception error) {
          throw error;
         }
         finally {
          if (rs != null) rs.close();
          if (stmt != null) stmt.close();
          if (conexion != null) conexion.close();
         }
       return arregloLocalidad;
    }
}
```

La clase LocalidadDAO establece un método público que no recibe parámetros, que devolverá un ArrayList, que es un objeto que es capaz de contener información en forma de listado y este objeto forma parte del API de Java, por tanto se puede consultar la documentación del API para saber más de este objeto.

En la siguiente línea se esta creando un objeto tipo ArrayList con el nombre de arregloOrigen,

Un elemento esencial en un DAO es la construcción de la consulta (query) a la base de datos en la parte donde se esta declarando una variable (sql) tipo cadena (String) y le asigna a ésta el valor del query por tanto el query estará almacenado en la variable sql.

En el bloque ***try*** se manda llamar al objeto conexiónDB y ejecuta el método getConexion() con el que se establece la conexión con la Base de datos.

El trato que se da a un query es a través de un objeto conocido como Statement el cual es creado por el objeto connection. La variable asignada como statement tendrá la capacidad de ejecutar el query con el método executeQuery(sql), obsérvese que el parámetro que se le esta entregando al método executeQuery es la variable de tipo cadena establecida líneas atrás.

El resultado de la ejecución del método executeQuery() se almacena en un objeto ***Resultset*** para guardar en él la lista de localidades y claves.

El resultado será manipulado por un ciclo mediante una estructura de control while que se encargará de recorrer el objeto resultado con el método **next().**

Para que se comprenda mejor, el ***Resultset*** denominado ***resultado*** contiene una lista de n número de localidades cada una de ellas acompañada de su clave. Supongamos que en la primera posición se encuentra localidad1 – clave1. este resultado es extraído con el método next() pero necesitamos almacenarlo en el Arraylist con su método add() el cual hará uso de el TransferObject correspondiente que asocie este par de datos (localidad y clave) con sus correspondientes getters y de esta manera podamos cargar la información al TO y así quede constituido un Arraylist de TransferObjects que será devuelto a la JSP mediante la acción **return arregloLocalidad** que se encuentra justo antes de cerrar el método construido.

En el caso de que alguno de los pasos anteriores no logre ejecutarse con éxito se entrara en el bloque catch que lanzará un aviso de error, pero si todo ocurre exitosamente el bloque catch será ignorado.

Por último esta el bloque finally que se encarga de cerrar o destruir el Statement, el Resultset y la conexión con la base de datos.

4.9.6 Validador

Cuando una aplicación cliente-servidor es implementada es posible que el cliente intente interactuar con el servidor enviando datos incorrectos ya sea por descuido o desconocimiento de lo que la aplicación espera.

Una base de datos no valida información, simplemente la acepta o la rechaza de tal suerte que, si se le envía información que no corresponde con la que espera no se realizará cambio alguno y esto originara fallas o errores de algún tipo.

Para evitar lo anterior crearemos una clase intermedia que valide la información, es decir que la reciba, haga que coincida con una forma adecuada y le permita continuar con su trayecto hacia la base de datos, para así garantizar el correcto funcionamiento de nuestra aplicación.

CorreoValidador.java

```
public class CorreoValidador {
    public Object verificaDatos(Map pParametro)
    throws ValidaError {
       Object parametro = null;
       String valores[] = null;

       /*Extraemos un elemento del Objeto Map mediante el
        *método get()*/
       parametro = pParametro.get("nombre");

       /*Verificamos si el elemento es nulo, longitud < 1
        *o el valor de cadena es vacío (“”)*/
       if (parametro  == null ||
          (valores = (String[])parametro).length < 1 ||
           valores[0].trim().equals("")) {

          /**Si alguna de estas condiciones se cumple el dato
          * es erróneo y se envía el siguiente mensaje*/
          throw new ValidaError("Nombre Invalido");
       }

       parametro = pParametro.get("email");
       if (parametro  == null ||
          (valores = (String[])parametro).length < 1 ||
           valores[0].trim().equals("")) {
          throw new ValidaError("E-Mail Invalido");
       }

       /** En este punto se carga un TransferObject para que
        *    los  datos  estén  disponibles  para  las  siguientes
clases
        *  y así pueda ser transferida la información */

       CorreoTO correoTO = new CorreoTO();
       correoTO.setNombre (((String [])pParametro.get("nombre")) [0]);
       correoTO.setEmail  (((String [])pParametro.get("email"))  [0]);
    }
}
```

Es posible realizar cualquier tipo de verificación o manipulación que sea necesaria para que la información caiga correctamente en la base de datos, en el caso anterior, como sabemos que la base de datos no admite que estos campos sean nulos por lo tanto nos aseguramos que exista información en ellos, de lo contrario se genera un error y un mensaje manipulado por la clase ValidaError().

4.9.7 Proxy

Un Proxy es un componente intermedio entre el servlet y el DAO que nos ayudara a implementar la lógica de negocio ya que realizará las operaciones necesarias para ejecutar el DAO y manipular los resultados que provengan de este.

ConsultaInfoProxy.java

```
public class ConsultaInfoProxy{
    InfoDAO infoDAO;

    public InfoTO realizaConsulta (registroTO pReg) throws
    Exception{
        InfoTO infoConsulta = new InfoTO(pReg.getName(),
pReg.getMail());
        infoDAO = new InfoDAO();

        infoConsulta = infoDAO.ejecutaConsulta(infoConsulta);

        return infoConsulta;
    }

}
```

En esta clase podemos apreciar que se está implementado un método que se propone recibir como parámetro un TransferObject el cual será utilizado para dar forma a otro tipo de TransferObject mediante el cual se podrá ejecutar un método propio del DAO el cual esta configurado para recibir como argumento el segundo tipo de TO, por ello es necesario dicho intercambio de información entre TransferObjects.

Este tipo de conductas las podemos implementar en un DAO, de hecho podemos manipular varios DAOs en el. Podemos relacionar en esta clase los TrasnsferObjects, con los DAOs con el fin de proveer el comportamiento que la aplicación necesite.

4.9.8 Socket

La aplicación que nos proponemos realizar hace uso de diferentes fuentes de datos, la primera es una base de datos que proporcionará información como la de las localidades, la segunda es aquella que se encarga de manejar todo el funcionamiento de la empresa, y la tercera que es un servicio que proporciona la banca para realizar las transacciones económicas. Las últimas dos no funcionan de una manera convencional y por tanto no podemos acceder a ellas mediante consultas tipo sql sino que debemos transformar nuestras peticiones de tal forma que podamos comunicarnos con estas entidades.

La transferencia y comunicación de la información la llevaremos a cabo mediante una clase tipo Socket, la clase de ejemplo ha sido tomado del sitio http://java.sun.com[3] y a continuación lo explicaremos.

Obsérvese que se esta implementando un método main() o método principal, es decir, que a partir de este método se comienza a ejecutar el programa o en este caso la clase, sin embargo para el tipo de aplicación que estamos desarrollando no la utilizaremos, podemos crear cualquier otro método, como se a mostrado anteriormente que adquiera la funcionalidad sugerida en esta clase de ejemplo.

Una vez que se declaran los objetos que utilizaremos entramos al bloque try que define los 3 elementos básicos de un Socket, primeramente se crea un objeto Socket y se le pasan como parámetros el nombre de red del equipo al que se va a conectar y el puerto de comunicación en el que se establecerá la comunicación. El segundo elemento nos permitirá la escritura a través de un objeto PrintWriter que enviara la información al equipo referenciado a través de un Stream[4] de salida (getOutputStream). El tercer elemento es un objeto BufferedReader que nos permitirá la lectura de información proveniente del otro equipo mediante un stream de entrada (InputStreamReader).

[3] http://java.sun.com/docs/books/tutorial/networking/sockets/readingWriting.html
[4] Stream: Con esto nos referimos al flujo de información, ya que en Java se usa este tipo de elementos como los métodos de entrada y salida de datos como InputStreamReader y getoutputStream respectivamente.

EchoClient.java

```
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws IOException {

    Socket echoSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;

    try {
       echoSocket = new Socket("taranis", 7);
       out = new PrintWriter(echoSocket.getOutputStream(),true);
       in = new BufferedReader(new InputStreamReader(
       echoSocket.getInputStream()));
       } catch (UnknownHostException e) {
          System.err.println("Don't know about host: taranis.");
          System.exit(1);
        } catch (IOException e) {
          System.err.println("Couldn't get I/O for "
                             + "the connection to: taranis.");
          System.exit(1);
        }
    BufferedReader stdIn = new BufferedReader(
                           new InputStreamReader(System.in));
    String userInput;
    while ((userInput = stdIn.readLine()) != null) {
       out.println(userInput);
       System.out.println("echo: " + in.readLine());
    }

    out.close();
    in.close();
    stdIn.close();
    echoSocket.close();
    }
}
```

El primer bloque catch atrapa una excepción (error) UnknownHostException que ocurre cuando el Socket no puede establecer comunicación con el equipo definido a causa de que este no este disponible.

El segundo bloque catch previene una falla en la comunicación en caso de no poder realizar las acciones de lectura o escritura.

La clase continua ejecutándose en caso que no se ejecuten los bloques catch. Se entra en un proceso de lectura y manipulación del stream de entrada (proveniente del equipo remoto) y se imprime el resultado en pantalla.

Finalmente se cierran los objetos de lectura y escritura y el Socket.

Como podemos apreciar el Socket hace posible la interacción remota con equipo en la red proporcionando el nombre del equipo y un puerto de comunicación con el fin de intercambiar información mediante el envió y recepción de datos en forma de Streams.

4.9.9 Session Façade

El problema existente en ciertos sistemas distribuidos es que algunos elementos de la lógica de negocio pueden quedar expuestos en la capa cliente haciendo vulnerable nuestro sistema, además de que inevitablemente existe un alto acoplamiento entre el cliente y los datos de negocio.

El objeto Façade debe ser entendido como una fachada que se establece entre el cliente y la lógica de negocio. La razón de ser esta fachada es servir como un controlador del flujo de información con el motivo de proveer seguridad a la aplicación al momento de encapsular la lógica de negocio abstrayendo su complejidad y que esta no quede expuesta en la capa cliente, por otro lado permite reducir el alto acoplamiento ya que éste flujo manipulará los datos y procesos sin la necesidad de que el cliente intervenga, aminorando así, la dependencia del cliente con esta información.

De acuerdo con los requerimientos que la capa cliente exija será la manera de implementar la fachada, pero podemos aseverar que el objeto Façade funciona como objeto constitutivo de la capa de negocio que controla las interacciones entre la capa cliente y la capa de lógica de negocio.

De acuerdo con lo anterior se estila decir que el objeto Façade introduce la capa controladora para la capa de negocio.

En resumen, un objeto de negocio debe ser responsable de su propio control y las interacciones que éste tenga con otros objetos de negocio necesita ser controlada por un objeto conducido por el patrón Façade.

En consecuencia la implementación de este patrón ofrece el beneficio de reducir el flujo de información que fluye en la red optimizando así su funcionamiento, debido a que el cliente no interactúa directamente con los datos de negocio.

VentaSessionFacade.java

```
public class VentaSessionFacade {

    /* Este es un método que invoca el Proxy enviando los
     * datos de la tarjeta de crédito en un Transfer Object */

    public VentaTO ejecutaVenta(TarjetaTO pTarjetaTO)
                                          throws VentaError {

       VentaTO                datosVenta   =  null;
       RespuestaServicioTO    respuesta    =  null;

       /* El Bloque try invoca al servicio que provee el banco
        * y le entrega la información de la tarjeta de crédito,
        * el objeto respuesta obtiene el resultado del banco.
        * El resultado es manipulado y devuelto al Proxy     */

       try {
          respuesta = new Banco().DireccionaBanco(pTarjetaTO);
          datosVenta = (VentaTO) respuesta.getResultado();

       } catch (SQLException e) {
          System.out.println ("Tarjeta Invalida", e.getError());
       }
       return datosVenta;
    }
}
```

El boque catch sólo es ejecutado en el caso que ocurra una Excepción en el servicio del Banco ocasionado por datos incorrectos por parte del cliente.

Obsérvese que en este ejemplo conviven otros objetos como TransferObjects, Proxy, etc., los cuales los estamos haciendo interactuar con el fin de manejar toda la información desde el servidor sin que intervenga el cliente, desde que éste envía la información y hasta que se le entrega una respuesta.

Capítulo 5

CONSTRUCCIÓN DE UNA APLICACIÓN WEB QUE PERMITA VENDER Y RESERVAR BOLETOS DE AUTOBÚS MEDIANTE INTERNET

Entrando directamente a lo que es la construcción de aplicaciones Web tenemos que ver primero como se construye el ***frontend*** mediante el lenguaje HTML para posteriormente construir la lógica del negocio a través lenguaje Java.

5.1 Construyendo la Página Principal del Sitio

Considerando que una página puede ser tan elaborada como la imaginación lo permita, con imágenes, videos, audio, etc., tenemos la intención de simplificar al máximo la página, con el propósito de enfocarnos el los temas sustanciales de la investigación y no en aquellos elementos visuales que sin duda darán mejor apariencia al sitio pero que no son propiamente el motivo de la investigación.

La página principal será aquella que se desplegará en el momento en que el visitante introduzca la dirección correspondiente a nuestro sitio y tiene la intención de ubicar al visitante en cuanto al propósito del sitio y los servicios que provee, adicionalmente debe permitir hacer uso del servicio de consulta de viajes a partir del cual se derivan el resto de los servicios.

index.html

```
<HTML>
<HEAD>
<TITLE> Pagina Principal </TITLE>
<LINK REL="Stylesheet" TYPE="text/css" HREF="./estilo.css">
</HEAD>
<BODY>
     <FORM METHOD=POST ACTION="">
     <H1> BIENVENIDO AL SITIO DE LA EMPRESA DE AUTO-TRANSPORTE FORANEO</H1>
     <H2> USTED PODRA CONSULTAR, RESERVAR Y COMPRAR BOLETOS DE AUTOBUS
         DESDE LA COMODIDAD DE SU HOGAR A PARTIR DE ESTE SITIO</H2>
     <H3>Sólo siga las Instrucciones a continuación y usted estará
         consultando la información del viaje que vaya a realizar</H3>
     <HR COLOR="#967D32">
     <P>Introduzca en las siguientes casillas la información solicitada para
         consultar viajes</P>
     <TABLE BORDER="0">
         <TR HEIGHT="150">
             <TD> Elija la Localidad de Donde Iniciara su Viaje<BR>
                 <SELECT NAME="origen">
                     <OPTION SELECTED>Seleccione Origen</OPTION>
                 </SELECT>
             </TD>
             <TD WIDTH="50"></TD>
             <TD ALIGN="RIGHT"> Elija la Localidad a Donde desea Arribar<BR>
                 <SELECT NAME="destino">
                     <OPTION SELECTED>Seleccione Destino</OPTION>
                 </SELECT>
             </TD>
         </TR> <TR>
             <TD COLSPAN="3">
                 <CENTER>
                         Fecha: <INPUT TYPE="text"
                         NAME="fecha"VALUE="DD/MM/AAAA"> <BR>
                 </CENTER>
             </TD>
         </TR> <TR>
             <TD COLSPAN="3" HEIGHT="150">
                 <CENTER><INPUT TYPE="submit" NAME="search"
                                     VALUE="CONSULTA VIAJES"></CENTER>
             </TD>
         </TR></TABLE>
     </FORM> </BODY>
</HTML>
```

El resultado obtenido del código fuente anterior cuando se ejecuta a través de un explorador Web es el siguiente:

A continuación se analizará y relacionará cada elemento de la pantalla con el código.

Los elementos que se encuentran antes de la etiqueta <BODY> son meramente informativos y no se vera reflejado su resultado en la página, a excepción de la etiqueta <LINK> ya que esta hace un llamado a una Hoja de Estilo (StyleSheet), que no es más que un archivo que configura los colores, tamaños, tipos de letra, etc., de tal forma que todas las páginas HTML que construyamos a continuación harán referencia a este archivo para que todas tengan una apariencia estándar.

Los elementos que se encuentran entre las etiquetas <H1>, <H2>, <H3> y <P> es únicamente texto y lo que las etiquetas proveen al texto es formato, como puede ser tamaño, color, tipo de letra, etc., estos parámetros pueden ser manipulados también desde la hoja de estilo.

Posteriormente está una etiqueta <TABLE> que es una tabla donde acomodaremos los elementos que vamos a utilizar. Como toda tabla ésta contiene Renglones y Columnas definidas por las etiquetas <TR> y <TD> respectivamente.

Dentro de la Tabla usamos 2 tipos de elementos que llamaremos controles ya que proveen a la página funcionalidad, de tal forma que nos permitirá interactuar a través de éstos con el cliente

El primer tipo esta representado por un par de listas que desplegarán un conjunto de opciones para elegir una de ellas, definido por la etiqueta <SELECT>, donde cada opción estará definida dentro de las etiquetas <OPTION>. El resultado que podemos apreciar en la pantalla número 1 lo apreciamos en los rectángulos blancos que muestran la leyenda Seleccione Origen y Seleccione Destino.

El segundo tipo es un control de tipo texto y es aquí donde el cliente podrá introducir información, en este caso una FECHA con el formato que se sugiere.

En la pantalla de ejemplo podemos apreciar una imagen con apariencia de botón que funciona como un enlace a otra página, es decir que está configurada de tal forma que, cuando se haga clic sobre ella el explorador mandara a llamar a la dirección que hace referencia, por ello a este tipo de herramienta se le conoce como enlace, liga, referencia o link en inglés. Ésta referencia es definida en el atributo HREF="#" de la etiqueta <A>, donde la dirección a la siguiente página deberá plasmarse en lugar del símbolo #. En este caso particular como no tenemos otra página a donde dirigirnos, escribimos el símbolo para únicamente recargar o rellamar a la actual.

Como podemos observar las casillas de “Seleccione Origen” y “Seleccione Destino” están vacías, es decir que no tienen definidos los puntos de origen y destino, estos pueden ser agregados directamente a la página HTML con sólo agregar una etiqueta de esta forma <OPTION>México</OPTION> y aparecerá en la página como una opción de la lista desplegable, sin embargo esto acarrea un problema de mantenimiento de la página que se generará en el momento en que introduzca un nuevo origen o destino en la empresa ya que se tendrá que alterar la página HTML cada vez que esto ocurra.

La solución al problema anterior es posible si la información de origen y destino es recabada a partir de una base de datos que proporcione la información de manera actualizada al sitio. De este modo cuando se agreguen orígenes y destinos nuevos estos cambios se realizarán en la base de datos y no directamente en la página HTML.

Para que esto sea posible necesitamos un conjunto de clases Java que realicen los procesos necesarios para que sea extraída la información requerida de la base de datos y posteriormente se entregue la información recabada a la página HTML por medio del código Java que será incrustado directamente a esta página.

5.2 Construcción de la Aplicación

En el capítulo 3 hablamos ampliamente de lo que la arquitectura Java propone y los componentes más generales que la integran, complementariamente en este capítulo hablaremos más específicamente acerca del lenguaje de programación, como se utiliza de manera general a partir de sus componentes, del API y de ciertos patrones de diseño que nos ayudaran a darle una estructura sólida a partir de la cual podremos crear una aplicación Web.

En el capítulo anterior se plantea una solución general a partir de la cual podremos asociar los diversos componentes y patrones de diseño para construir la solución o mejor dicho la aplicación.

Para el problema planteado respecto de los orígenes y destinos indefinidos necesitamos extraer la información de la tabla Localidad que está alojada en la base de datos, esta tabla le proporcionará la información necesaria a las listas desplegables para mostrar las localidades y aquella información que será utilizable por el sitio.

Aquí es donde la situación se comienza a complicar. Para empezar la JSP necesita invocar un método que le permita solicitar la información a la base de datos.

El método al que nos referimos no existe aún, pero sabemos que debe ser parte de un una clase que este regida bajo el patrón de diseño conocido como DAO, el cual, como ya hemos explicado es un objeto que permite el acceso a datos, por tanto nos abocaremos a la creación de esta clase y el método correspondiente.

Antes de entrar a la construcción del DAO demos una ojeada a la tabla Localidad y reflexionemos acerca de aquellos datos que requerimos. Sabemos que lo que necesitamos de entrada es mostrarle al usuario el nombre de las localidades que serán los Orígenes y los Destinos y esta información, si observamos, esta contenida en el campo loc_nombre, podría bastarnos con esto, pero, también vemos un campo llamado loc_clave nos es útil para que cuando el cliente seleccione la localidad podamos regresar a hacer una búsqueda de los viajes usando la clave de localidad correspondiente a la selección hecha.

Con esta información podemos idear una consulta a la base de datos mediante un “Query” de SQL, que nos devolverá una lista con todos los registros.

La forma del Query seria la siguiente:

```
SELECT loc_nombre, loc_clave FROM localidad;
```

Con esta sentencia o query o consulta le estamos pidiendo a la base de datos que de la tabla localidad nos devuelva la información correspondiente a loc_nombre y loc_clave contenida en todos los registros.

La cuestión está en cómo podemos hacer para que el sistema automáticamente haga esta consulta a la base de datos a partir de la JSP y como manejaremos los registros que entregue dicha consulta para que estos sean mostrados al cliente a través de la JSP.

5.2.1 Página Principal convertida en JSP

Existen diversas formas de acceder a la información mediante una JSP debido a ésta puede hacer uso de código Java a través del cual podremos conseguir el objetivo. De hecho desde una JSP se puede obtener conexión directa a una base de datos y a su información, sin embargo la forma que propondremos tiene el objetivo, de ocultar al máximo los componentes de negocio por razones de seguridad, de tal forma que, aquello que la JSP maneje sea sólo información, ya que las consultas, conexiones y validaciones se manejaran mediante clases Java controladas a través de un Servlet.

La siguiente JSP no es más que la página HTML construida en el tema anterior a la cual le incrustaremos el código Java en 2 secciones principalmente, los controles de listas desplegables y en el atributo "ACTION" de la etiqueta <FORM>.

index.jsp

```
<HTML>
<HEAD>
<TITLE> Pagina Principal </TITLE>
<LINK      REL="Stylesheet"      TYPE="text/css"      HREF="C:\jakarta-tomcat-
4.1.12\utilerias\estilo.css">

<%@ page import="java.util.ArrayList" %>
<%@ page import="sitio.TO.LocalidadTO" %>
<%@ page import="sitio.dao.BuscaLocalidadDAO" %>

</HEAD>

<BODY>
     <FORM NAME="inicio" METHOD=POST
         ACTION="<%= request.getContextPath() %>/Search">
     <H1> BIENVENIDO AL SITIO DE LA EMPRESA DE AUTO-TRANSPORTE FORANEO</H1>
     <H2> USTED PODRA CONSULTAR, RESERVAR Y COMPRAR BOLETOS DE AUTOBUS
          DESDE LA COMODIDAD DE SU HOGAR A PARTIR DE ESTE SITIO</H2>
     <H3>Sólo siga las Instrucciones a continuación y usted estará
          consultando la información del viaje que vaya a realizar</H3>
     <HR COLOR="#967D32">
     <P>Introduzca en las siguientes casillas la información solicitada para
         consultar viajes</P>
     <TABLE BORDER="0">
         <TR HEIGHT="150">
             <TD> Elija la Localidad de Donde Iniciara su Viaje<BR>
                 <SELECT NAME="origen">
                     <OPTION SELECTED>Seleccione Origen</OPTION>
                     <%
                         BuscaLocalidadDAO traeInfo = new BuscaLocalidadDAO();
                             ArrayList arreglo = traeInfo.buscaLocalidades();
                             for (int i = 0 ; i < arreglo.size() ; i++) {
                                 LocalidadTO locacionTO
                                                 = (LocalidadTO)arreglo.get(i);
                     %>
                     <OPTION value="<%= locacionTO.getClave() %>">
                         <%= locacionTO.getNombre() %>
                     </OPTION>
                 </SELECT>
             </TD>
             <TD WIDTH="50"></TD>
             <TD ALIGN="RIGHT"> Elija la Locallidad a Donde desea Arribar<BR>
                 <SELECT NAME="destino">
                     <OPTION SELECTED>Seleccione Destino</OPTION>
                     <%
                         BuscaLocalidadDAO traeLoc = new BuscaLocalidadDAO();
                             ArrayList arregLoc = traeLoc.buscaLocalidades();
                             for (int i = 0 ; i < arregLoc.size() ; i++) {
                                 LocalidadTO locTO
                                             = (LocalidadTO)arregLoc.get(i);
                     %>
                     <OPTION value="<%= locTO.getClave() %>">
                         <%= locTO.getNombre() %>
                     </OPTION>
                     <%
                             }
                     %>
                 </SELECT>
```

```
            </TD>
        </TR>
        <TR>
            <TD COLSPAN="3">
            <CENTER>    Fecha:
                <INPUT TYPE="text" NAME="fecha" VALUE="DD/MM/AAAA"><BR>
            </CENTER>
            </TD>
        </TR>
        <TR>
            <TD COLSPAN="3" HEIGHT="150">
            <CENTER>
                <INPUT TYPE="submit" NAME="search" VALUE="CONSULTA VIAJES">
            </CENTER>
            </TD>
        </TR>
    </TABLE>
    </FORM>
</BODY>
</HTML>
```

Compare esta página JSP con index.html y se dará cuenta que en esencia es la misma, con ciertas adiciones hechas al haber incrustado código Java el cual podemos localizar entre estas etiquetas <% %>.

Es necesario saber que la ejecución de una JSP no es tan simple como una página HTML puesto que la primera requiere residir en la estructura de archivos creada para este caso dentro de el contenedor WEB que hayamos instalado en nuestro equipo de desarrollo además de que deben existir las clases que aquí se solicitan mediante la sentencia import al inicio de la página.

```
<%@ page import="sitio.TO.LocalidadTO" %>
<%@ page import="sitio.dao.BuscaLocalidadDAO" %>
```

Lo anterior sugiere que en nuestro proyecto existen y funcionan las dos clases a las que se esta haciendo referencia, sin embargo, es una tarea a la que nos avocaremos en los próximos temas, lo importante aquí es visualizar como es la estructura de la JSP y entender su funcionalidad.

Volviendo al tema del contenedor Web, en este caso particular haremos uso de APACHE TOMCAT que como ya comentamos en capítulos anteriores es una herramienta de uso generalizado, estable y en constante desarrollo.

Para instalar este software en nuestro equipo debemos primero descargarlo desde la página de APACHE Project. Ahí se podrán encontrar varias versiones, para este caso específico, utilizamos la versión jakarta-tomcat-4.1.12, la cual viene en un archivo comprimido (.zip).

5.2.1.1 Instalación y configuración del contenedor Web jakarta-tomcat-4.1.12

1. Para instalarlo únicamente tenemos que descomprimirlo en alguna ubicación de nuestro sistema de archivos, recomendamos descomprimirlo en la unidad C:/ de nuestro sistema para ubicarlo fácilmente cuando hagamos uso de él.

2. La configuración consiste únicamente en crear una NUEVA Variable de Sistema con el nombre de CATALINA_HOME y con el valor correspondiente a la ubicación donde reside el contenedor WEB, que en nuestro caso será C:\jakarta-tomcat-4.1.12. La variable de Sistema se configura en la sección variables de entorno que se localiza en el elemento Sistema del Panel de Control, en la etiqueta de Opciones Avanzadas

Debe tenerse en cuenta que de acuerdo a la versión en la manera de instalación, aunque cabe destacar que versiones más recientes cuentan con un instalador que facilita este proceso.

Ya que se ha instalado el contenedor Web podemos continuar con la creación del sistema de archivos que contendrán la aplicación que nos proponemos desarrollar.

1. Al nivel de la carpeta C:\jakarta-tomcat-4.1.12 crearemos una carpeta que contenga todo el contenido no dinámico de nuestro proyecto, como pueden ser páginas HTML, Hojas de Estilo e imágenes. En este caso lo llamaremos utilerías.

2. Estando ubicados en la carpeta C:\jakarta-tomcat-4.1.12 podremos localizar una carpeta llamada webapps que se refiere a las Aplicaciones Web (WebApplicatios) que residen en el contenedor, de tal suerte que cada carpeta que aquí se encuentra representa una aplicación. En este nivel crearemos una carpeta que represente la aplicación Web que nos proponemos desarrollar, para este caso específico le llamaremos \tesis.

3. esta carpeta deberá contar indispensablemente con una carpeta llamada \WEB-INF que contiene un archivo de configuración XML. Recomendamos que se dirija a la carpeta ROOT la cual contiene la carpeta WEB-INF la seleccionemos creemos una copia de esta carpeta en la de nuestro proyecto tesis y así poder contar con la estructura básica indispensable para que nuestra aplicación funcione.

4. Por último es necesario crear una carpeta \classes dentro de la carpeta WEB-INF la cual contendrá las clases que darán funcionalidad a nuestra JSP. Esto será explicado a detalle más adelante.

Todo lo anterior es necesario debido a que en este sistema de archivos será donde deberá estar colocado el archivo index.jsp que más específicamente debe residir en la carpeta tesis. No debe perderse de vista que index.jsp desencadenara un comportamiento tal que hará uso de los recursos que residen en el proyecto tales como las clases, las utilerías, y el archivo de configuración web.xml que son los elementos constitutivos del proyecto que desarrollaremos.

Lo que requerimos para que logremos incrustar las localidades en las listas desplegables de la JSP, como podemos apreciar en las sentencias import es un DataAccesObject que busque y regrese las localidades y un TransferObject que contenga y transfiera la información de cada localidad que será entregada a la JSP.

5.2.2 LocalidadTO

El Transfer Object que aquí requerimos es muy simple debido a que sólo utilizaremos dos datos el nombre y la clave de la localidad, por lo tanto su forma es la siguiente:

LocalidadTO.java

```
package sitio.TO;

public class LocalidadTO {
    private String clave;
    private String nombre;

    public LocalidadTO(){}
    public LocalidadTO(String pClave, String pNombre){
       clave = pClave;
       nombre = pNombre;
    }

    public String getClave() {
       return clave;
    }
    public String getNombre() {
       return nombre;
    }
    public void setClave(String clave) {
       this.clave = clave;
    }
    public void setNombre(String nombre) {
       this.nombre = nombre;
    }
}
```

Como pedemos apreciar la clase lleva el nombre de LocalidadTO y cuenta con dos variables privadas de tipo String correspondientes a los dos datos que posteriormente extraeremos de la base de datos.

Enseguida encontramos los constructores de la clase mediante los cuales podremos crear instancias a esta clase para hacer uso de ella.

Posteriormente hacemos uso de lo que conocemos como setters, que son los métodos que utilizaremos para introducir la información en el TO o dicho en términos de desarrollo cargar el TO y los getters que son los métodos mediante los cuales obtenemos la información que contiene el TO para hacer uso de ella.

En el DAO podremos apreciar como se utilizan los setters y en la JSP se utilizaran los getters.

El DAO como tal, típicamente realiza el acceso a una base de datos, sin embargo para que esto sea posible primero requerimos establecer una conexión con dicha base de datos y para ser consecuentes con el paradigma de programación orientada a objetos debemos de crear de manera independiente una clase que se encargue de dicha tarea en específico.

5.2.3 ConexiónDB

Antes de entrar en materia de construcción de la clase Conexión debemos crear un Nombre de Origen de Datos o DSN (por sus siglas en inglés) para que sea posible la conexión mediante el controlador ODBC.

Para llevar a cabo lo anterior es necesario seguir estos pasos.

1. Entrar al Panel de Control → Herramientas de Sistema → Orígenes de Datos (ODBC).

2. Ubicarse en la pestaña DSN Usuario y hacer clic en el botón agregar.

3. Elegir en controlador correspondiente a la base de datos a la que se establecerá el origen de datos.

4. Aparecerá una ventana que pedirá ciertos datos, dependerá del tipo de controlador elegido la información solicitada pero en esencia se debe dar un nombre de origen de datos, seleccionar la ubicación de la base de datos, proveer un nombre de usuario y una contraseña.

5. Para finalizar simplemente hacemos clic en el Botón ACEPTAR y el DSN ha sido creado.

Ahora si podemos dar paso a la creación de la clase conexión.

La clase que a continuación mostramos implementa el tipo de conexión más simple que es el de Puente JDBC-ODBC que para fines de esta investigación es suficiente aunque para casos más avanzados recomendamos hacer uso de otros tipos de controladores JDBC para que el desempeño de la aplicación sea óptimo. Pero debemos hacer notar que el sentido de hacer una clase conexión se realiza de manera independiente al DAO porque cuando se decida cambiar de controlador JDBC sólo se realizaran cambios en esta clase y el resto de las clases permanecerán intactas y su funcionamiento no será alterado en absoluto.

ConexionDB.java

```
package sitio.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexionDB {
    public ConexionDB(){
       cargaControlador();
    }
    private void cargaControlador() {
       String controladorBD = "sun.jdbc.odbc.JdbcOdbcDriver";
       try{
          Class.forName(controladorBD);
       }catch (ClassNotFoundException error){
          System.out.println("Error Cargando Puente ODBC-JDBC");
       }

    }
    public Connection getConexion()throws Exception {
       String direcciona = "jdbc:odbc:sitiodb";
       String usuario = "system";
       String contrasenia = "contrasenia";

       return  DriverManager.getConnection(direcciona,  usuario,
contrasenia);
    }

    public void cierraConexion (Connection pCon) {
       try {
          if (!pCon.isClosed()) {
             pCon.close();
          }
       } catch (SQLException e) {
          e.printStackTrace();
       }
    }
}
```

Observemos que la clase ConexionDB tiene un constructor que ejecuta el método cargaControlador(), donde lo importante ocurre en el bloque try donde se manda a llamar el controlador con esta cadena sun.jdbc.odbc.JdbcOdbcDriver mediante el método Class.forName de esta forma se establece el puente JDBC-ODBC que nos permitirá interactuar con la base de datos.

Cuando lo anterior ocurre exitosamente desde el DAO se llamará el método getConexion() que se encargara de establecer la conexión con la base de datos pasándole al método getConection los argumentos correspondientes al nombre del origen de datos, el nombre de usuario y la contraseña de la base de datos. Si estos elementos son correctos se establecerá exitosamente la conexión con la Base de datos y se podrá acceder a los recursos que esta ofrece.

5.2.4 BuscaLocalidadDAO

A continuación mostramos la clase BuscaLocalidadDAO encargada de extraer de la base de datos el nombre y clave de las localidades en el sistema.

BuscaLocalidadDAO.java

```
package sitio.dao;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

import sitio.TO.LocalidadTO;
import sitio.database.ConexionDB;

public class BuscaLocalidadDAO {
    public ArrayList buscaLocalidades()throws Exception{
       ArrayList localidades = new ArrayList();
       ConexionDB connDB = new ConexionDB();
       Connection conecta = null;
       Statement sentencia = null;
       ResultSet respuesta = null;


       String sql;
       sql = "SELECT loc_clave, loc_nombre FROM localidad";
       try{
          conecta = connDB.getConexion();
          sentencia = conecta.createStatement();
          respuesta = sentencia.executeQuery(sql);
          while(respuesta.next()){
             localidades.add(new LocalidadTO
                              (respuesta.getString("loc_clave"),
                           respuesta.getString("loc_nombre")));
          }
       }catch(Exception error){
          throw error;
       }
       finally{
          if(respuesta!=null)
             respuesta.close();
          if(sentencia!=null)
             sentencia.close();
          if(conecta!=null)
             conecta.close();
       }
       return localidades;

    }
}
```

Podemos apreciar que esta clase implementa un método buscaLocalidades() el cual, no contiene argumentos pero que si devuelve un ArrayList esto es porque cuando la ejecutemos desde la JSP requerimos que a la JSP le sea entregado un listado con las Localidades existentes, las cuales serán almacenadas en esta estructura de Java denominada ArrayList.

En el bloque inicial se hacen las instancias correspondientes a los objetos ArrayList y ConexionDB y se crean algunas variables.

Como observamos vamos a hacer uso de la clase que acabamos de crear. La variable de tipo Statement auxilia en el llamado y ejecución del Query que definimos en la variable "sql". Esto nos traerá como resultado un conjunto de elementos que serán almacenados en un ResultSet que es la manera en que los resultados provenientes de una base de datos son obtenidos.

Este conjunto de resultados o ResultSet será manipulado a través de una estructura de control While, lo que permitirá extraer uno a uno los datos contenidos en él, almacenarlos en un Transfer Object y organizarlos en un ArrayList, que como ya comentamos será la manera en que se devolverá la información a la JSP.

Si todo ocurre satisfactoriamente y sin contratiempos se efectúa el bloque finally que se encarga de cerrar el Statement, Resultset y la conexión que por razones de seguridad deben permanecer cerrados si no se están utilizando.

5.2.5 La JSP en Funcionamiento

Una vez entendido lo anterior podemos concluir la explicación del funcionamiento que provee el código Java en la JSP.

La JSP incluye este código asociado a los controles de listas desplegables.

```
<SELECT NAME="origen">
    <OPTION SELECTED>Seleccione Origen</OPTION>
       <%
          BuscaLocalidadDAO traeInfo = new BuscaLocalidadDAO();
          ArrayList arreglo = traeInfo.buscaLocalidades();
          for (int i = 0 ; i < arreglo.size() ; i++) {
          LocalidadTO locacionTO = (LocalidadTO)arreglo.get(i);
       %>
    <OPTION value="<%= locacionTO.getClave() %>">
       <%= locacionTO.getNombre() %>
    </OPTION>
</SELECT>
```

Ahora resulta más sencillo entender el porque de la situación aquí planteada, primero esta definiéndose un control de lista desplegable que en código HTML esta definido por el SELECT, cada una de las opciones presentadas será establecida por la etiqueta OPTION.

Entramos al bloque de código JSP y encontramos que se esta creando una instancia del Objeto BuscaLocalidadDAO() y se esta haciendo uso de su método buscaLocalidades() que se espera que devuelva un ArrayList, por esta razón se esta igualando el objeto ArrayList con el resultado de devenga del método invocado.

Posteriormente se recorre el arreglo para extraer uno a uno los elementos que contenga y se almacenan en el Transfer Object correspondiente.

Finalmente extrae el dato contenido en el Transfer Object mediante el uso de sus getters para plasmar la información obtenida en la etiqueta OPTION correspondiente y de esta manera obtener un resultado de este tipo.

Lo que hemos hecho hasta el momento es la creación del código en clases java, sin embargo para que esta JSP funcione requiere hachar mano de estas clases que hemos creado y para que tal cosa ocurra necesitamos compilar estas clases. Al compilar cada clase se genera un archivo con extensión .class el cual queda organizado de acuerdo al package que se le asigna.

Obsérvese que en la primera línea de cada clase se declara un package que es un paquete en el que organizamos el proyecto que estamos definiendo, es como una estructura arbórea de carpetas donde ubicamos las clases de acuerdo a su función o tipo.

La raíz de todos los paquetes que generemos será un paquete que nombraremos “sitio”. Una vez generados los archivos .class contenidos en nuestro paquete sitio, que a nivel de archivos es una carpeta del mismo nombre a la cual le haremos una copia y la posicionaremos en la carpeta class localizada en el WEB-INF de nuestra aplicación residente en el Contenedor Web.

Para echar a andar la aplicación primeramente necesitamos levantar el TOMCAT, es decir ejecutar un archivo que activara el servicio que proporciona el contenedor Web. Para esto es necesario ejecutar el archivo startup.bat ubicado en C:\jakarta-tomcat-4.1.12\bin\startup.bat.

A continuación se debe ejecutar un explorador de Internet o Web Browser e introducir la siguiente dirección http://localhost:8080/tesis/index.jsp. Donde http se refiere al protocolo de hipertexto, el localhost se refiere a el equipo desde el cual se ejecutara la aplicación (en este caso particular el propio equipo de desarrollo), :8080 se refiere al puerto que asigna tomcat para entregar la información, tesis se refiere a la ubicación de nuestra aplicación Web y claro esta que index.jsp es la que acabamos de crear.

Si no se registra error alguno el resultado se apreciara como en la siguiente imagen:

Obsérvese que al hacer clic en la lista desplegable Seleccione Origen se muestran 5 localidades que han sido extraídas directamente de la base de datos, a esto es a lo que llamamos un comportamiento dinámico de una página WEB de tal suerte que cualquier cambio que se efectúe en la tabla localidades se vera reflejado al instante cuando sea llamada esta página principal.

Cuando se haga clic en el botón Consultar se ejecutará el evento ACTION de la etiqueta <FORM> de la JSP la cual es una sentencia JSP que direccionará la consulta hacia un Servlet.

5.3 Consulta de Viajes

Al momento de hacer clic en el botón consulta se ejecuta un Servlet e cual desencadena todo un comportamiento que hace uso de varias clases que obtienen información referente a los viajes que se ajusten a las características definidas por el cliente en la página de inicio, esta acción vista desde el equipo cliente puede demorar unos cuantos segundos pero lo que implica en desarrollo es bastante complejo, a explicar todo lo que hay detrás de esto nos abocaremos a continuación.

5.3.1 ListadoViajesServlet

Si observamos en el direccionamiento efectuado por el elemento ACTION de la JSP, este remite al elemento Search, el cual es probable que no reconozcamos, esto se debe a que un Servlet es puesto a disposición de la aplicación mediante el archivo de configuración WEB.XML donde encontraremos este elemento Search al que nos referimos.

```
                                    WEB.XML
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN">
<web-app>
     <display-name>Welcome to Tomcat</display-name>
     <description>
         Welcome to Tomcat
     </description>
     <servlet>
         <servlet-name>ListadoViajesServlet</servlet-name>
         <servlet-class>sitio.servlet.ListadoViajesServlet</servlet-class>
     </servlet>
     <servlet-mapping>
         <servlet-name>ListadoViajesServlet</servlet-name>
         <url-pattern>/Search</url-pattern >
     </servlet-mapping>
</web-app>
```

Como podemos apreciar esta URL esta asociada a Servlet denominado ListadoViajesServlet que es la clase que nos disponemos a explicar.

ListadoViajesServlet.java

```
package sitio.servlet;

public class ListadoViajesServlet extends HttpServlet {

    public void doPost(HttpServletRequest peticion,
           HttpServletResponse respuesta)throws ServletException{
       Object infoValida = null;
       infoValida = validaInfoViaje(peticion.getParameterMap());
       procesaPeticion(infoValida, peticion, respuesta);
    }

    public Object validaInfoViaje(Map pParam){
       InfoViajeValidador validando = new InfoViajeValidador();
       Object infoValida = validando.validadorInfo(pParam);
       return infoValida;

    }

    public void procesaPeticion (Object infoValida,
    HttpServletRequest peticion, HttpServletResponse respuesta){
       ViajeTO infoViajeTO = (ViajeTO) infoValida;
       ArrayList listadoViajes = null;
       ListadoViajesProxy peticionProxy = new ListadoViajesProxy();

       try {
           listadoViajes = peticionProxy.BuscaViaje(infoViajeTO);
           peticion.setAttribute("Viajes", listadoViajes);
           ServletContext traeContexto = getServletContext();
           RequestDispatcher envia = traeContexto.
                        getRequestDispatcher("/ListadoViajes.jsp");
           envia.forward(peticion,respuesta);
       } catch (ServletException e) {
           System.out.println("Excepcion en el Servlet: " + e);
           e.printStackTrace();
       } catch (IOException e) {
           System.out.println("Excepcion Entrada/Salida: " + e);
           e.printStackTrace();
       }
}
```

Existen 2 métodos mediante las cuales una JSP se comunica con un Servlet, a través del método Get o Post, para este caso particular, como conocemos que la JSP implementa el método Post el Servlet estará preparado para interactuar con la JSP a través del método doPost().

El método doPost desencadena el funcionamiento del Servlet, por tanto desde aquí se manipula la información que llega de la JSP a través de HttpServletRequest y se direcciona hacia una clase que llamaremos Validador que se encargará de comprobar que la información introducida por el cliente sea correcta, si esto es así, la información validada regresa al Servlet y se pone a disposición de otro método llamado procesaPeticion que direccionará la información hacia una clase tipo Proxy que hará las diligencias necesarias para obtener el listado de viajes que necesitamos para retornar la información a través del método getRecuestDispacher el cual direccionará el resultado a la JSP encargada de mostrar la respuesta recabada a través de la petición realizada.

En esencia un Servlet únicamente direcciona y canaliza el flujo de la información ya que los elementos que realmente desempeñan un proceso complejo son los que manda a ejecutar el Proxy.

5.3.2 InfoViajeValidador

Esta clase es muy simple pero es importantísima, ya que evitará que información errónea llegue a la base de datos, lo cual pondría generar problemas en la consistencia de la información.

```
package sitio.validador;
import java.util.Map;
import sitio.TO.ViajeTO;

public class InfoViajeValidador {
    public ViajeTO validadorInfo(Map pParam){
       ViajeTO viajeTO;
       Object pOrigen;
       pOrigen = pParam.get("origen");
       Object pDestino;
       pDestino = pParam.get("destino");
       Object pFecha;
       pFecha = pParam.get("fecha");
       if(pFecha == null){
          System.out.println("Error en la Fecha");
       }
       viajeTO = new ViajeTO();
       viajeTO.setOrigen(pOrigen.toString());
       viajeTO.setDestino(pDestino.toString());
       viajeTO.setFecha(pFecha.toString());
       return viajeTO;
    }
}
```

Tres cosas ocurren principalmente en esta clase, primeramente se obtiene la información proveniente de la JSP por medio del Objeto Map que aquí se identifica por pParam, al cual se le extraerá la información que provenga de los controles de la JSP, pParam.get le esta pidiendo extraer la información proveniente de un control, específicamente del dato que se encuentra entre paréntesis (“origen”) que si se observa la JSP corresponderá a el nombre de la lista desplegable Seleccione Origen.

Una vez extraído el dato, este es susceptible a ser manipulado, verificado, etc. Para darle la forma en que nos es útil el dato.

Por último los datos, ya extraídos y verificados serán introducidos en un TransferObject (ViajeTO) que permitirá trasladar la información conjunta que acabamos de recopilar y enviarla en esta forma hacia las demás clases que usaran esta información.

5.3.3 ListadoViajesProxy

Este elemento nos auxiliara en desmenuzar la complejidad de la sección que nos proponemos a elaborar puesto que es aquí donde la información es puesta a disposición del resto de las clases que se encargan de proveer la lógica de negocio que se va a implementar y además por este mismo medio se entregan las respuestas que serán devueltas al Servlet.

ListadoViajesProxy.java

```
package sitio.proxy;

import java.util.ArrayList;

import sitio.TO.ViajeTO;
import sitio.dao.ViajeDAO;

public class ListadoViajesProxy {
    public ArrayList BuscaViaje(ViajeTO infoViajeTO){
       ArrayList listadoViajes = null;
       ViajeDAO viaje = new ViajeDAO();
       listadoViajes = viaje.buscaViajeDAO(infoViajeTO);
       return listadoViajes;

    }

}
```

Este Proxy es muy simple puesto que únicamente crea un ArrayList el cual recibirá el resultado proveniente del método buscaViajeDao() de la clase DAO Viaje, como podemos observar este método devuelve un ArrayList que es precisamente el dato que se esta esperando en el Servlet al ejecutar este método.

5.3.4 ViajeDAO

Podríamos decir que el elemento más importante es éste, ya que es el que puede proveer y obtener información a partir de los orígenes de datos en donde la empresa fundamenta su funcionamiento.

ViajeDAO.java

```
package sitio.dao;
import java.util.ArrayList;
import sitio.TO.ViajeTO;
import sitio.empresa.ConexionEmpresa;
import sitio.interfases.ResultSetEmpresa;

public class ViajeDAO {

    public ArrayList buscaViajeDAO(ViajeTO infoViajeTO) {
       ArrayList listadoViajes = new ArrayList();
       ConexionEmpresa  conectEmp = new ConexionEmpresa();
       ResultSetEmpresa resultadoEmp[] = null;
       resultadoEmp = conectEmp.executeQuery(infoViajeTO);
       for (int i = 0 ; i < resultadoEmp.length ; i++ ) {
          while (resultadoEmp[i].next()) {
             listadoViajes.add(new
                   ViajeTO(resultadoEmp[i].getString("origen"),
                         resultadoEmp[i].getString("destino"),
                         resultadoEmp[i].getString("fecha"),
                         resultadoEmp[i].getString("hora")
                   ));
          }
       }     return listadoViajes;
    }
}
```

Primeramente encontramos la creación de objetos que vamos a utilizar, el ArrayList va a almacenar la respuesta, El objeto ConexionEmpresa nos proveerá la entrada al origen de datos con el que vamos a interactuar, el ResultSet nos permitirá manipular la información que devuelva la consulta que se ejecute.

El método executeQuery es el que echará mano de estos elementos, establecerá la conexión, realizará la consulta, y posicionará el resultado en el ResultSet, Posteriormente el ResultSet será manipulado para extraer la información, colocar esta información en el TransferObject correspondiente y crear un ArrayList de TransferObjects que es el que será devuelto al Proxy.

Hasta aquí podemos ver un comportamiento muy similar al primer DAO que analizamos, el de LocalidadesDAO, la verdadera complejidad se encuentra en el elemento conexión, el cual deberá establecer un tipo de conexión con un sistema no homogéneo, es decir, un sistema que por principio de cuentas no es una base de datos como tal y que esta montado en una plataforma completamente diferente a la que estamos utilizando, que no es ni Windows, ni Linux, ni Macintosh, y por lo cual no podremos acceder a ella vía JDBC, sino que tendremos que establecer comunicación vía Socket.

Cuando hablamos de Sockets nos referimos a una clase que encargue de establecer comunicación remota con otro equipo, lo que nos permitirá implementar un método propio de consulta (executeQuery) para enviar o recibir datos.

Considerando que obtendremos una respuesta por este medio necesitaremos crear una estructura propia de respuesta, es decir, implantar una interfase preparada para recibir la información proveniente y que funcione como un ResultSet para que a partir de él podamos descomponer la información en un ArrayList tal y como lo hicimos en el la clase LocalidadesDAO, y así podamos devolver la información hacia el Proxy.

5.3.5 ConexionEmpresa

En contraste con la otra clase de conexión que habíamos construido, en ésta no se realizará una implementación de controladores JDBC en cambio se hará una petición de información mediante el método executeQuery, el cual construiremos y se encargará de proveer la información y dirigirla al Socket. Este método estará esperando una respuesta a dicha petición y como sabemos que esperamos un listado de viajes, será necesario depositar la respuesta en un ArrayList, posteriormente este objeto se encargará de componer el ResultSet que será el elemento que devolveremos al DAO.

La información que va a ser dirigida hacia el Socket debe sufrir una transformación, ya que el Socket se comunicará con un equipo remoto el cual sólo admite una cadena de caracteres y lo que hasta ahora tenemos es un TransferObject con información, por lo tanto, debemos extraer dicha información del TO, convertirla a tipo String y darle el formato que espera el equipo remoto. Ejemplo: viaje/0001/0002/24/12/07.

ConexionEmpresa.java

```
package sitio.empresa;

import sitio.TO.ViajeTO;
import sitio.interfases.ResultSetEmpresa;

public class ConexionEmpresa {

    public ConexionEmpresa(){
    }
    public ResultSetEmpresa[] executeQuery(ViajeTO infoViajeTO) {
       ArrayList arregloViajes = new ArrayList();
       try {
          String trama = "viaje/" +
                    infoViajeTO.getOrigen().toString() + "/" +
                    infoViajeTO.getDestino().toString() + "/" +
                    infoViajeTO.getFecha().toString();
          arregloViajes.add(FormandoResultSet.setResultados(new
                       EmpresaSocket().procesaPeticion(trama)));
       }catch (IOException e) {
          System.out.println("Error Entrada/Salida: " + e);
          e.printStackTrace();
       }

       ResultSetEmpresa resultSetViajes[] =
                    new ResultSetEmpresa[arregloViajes.size()];

       for (int i = 0 ; i < resultSetViajes.length ; i++) {
          resultSetViajes[i] = (ResultSetEmpresa) arregloViajes.get(i);
       }
       return resultSetViajes;
    }
}
```

Cuando incluimos los corchetes en resultSetViajes[] estamos indicando que lo utilizaremos este elemento a manera de arreglo, y en la línea donde se instancia podemos apreciar que en los paréntesis se le asigna el tamaño del arreglo mediante la petición del tamaño (size) del ArrayList construido. Finalmente el ciclo **for** recorre el ArrayList extrae la información elemento por elemento y lo va depositando en el arreglo del ResultSetViajes que como ya dijimos es el objeto que será devuelto a la clase DAO.

5.3.6 EmpresaSocket

El objetivo de esta clase es crear un enlace entre la aplicación y un equipo remoto, este otro equipo se encuentra en la misma red local por lo que es conocido su nombre dentro de la red o tiene una dirección ***ip*** conocida, por lo tanto esta dirección será proporcionada como el primero de los dos argumentos en el momento en que se realice la instancia al objeto Socket, el segundo argumento es un número de cuatro dígitos que se refiere al puerto de comunicación del equipo remoto, de esta forma se establece la comunicación y a través de este puerto entrará y saldrá la información requerida.

Debido a que el Socket es un elemento que permite la entrada y salida de información resulta necesario crear objetos que permitan el envió (PrinWriter) y recepción (BufferedReader) de este flujo de información.

El flujo de información ocurrirá en el método procesaPeticion(pTama) de la clase (obsérvese que el parámetro pTrama del método es la cadena de caracteres que construimos en la clase ConexionEmpresa), ya que es aquí donde la información cargada en el objeto PrintWriter es enviada al otro equipo, y de igual manera la respuesta obtenida será cargada en el BufferedReader, además de que el mensaje será convertido en una variable de tipo cadena que es la manera en que la información requiere ser devuelta a la clase que invoco al Socket.

EmpresaSocket.java

```
package sitio.empresa;

public class EmpresaSocket {
    private BufferedReader recibe;
    private PrintWriter    envia;
    private Socket         socketCliente;

    public EmpresaSocket(ViajeTO infoViajeTO) throws IOException
{
        try {
           socketCliente = new Socket("125.45.45.1", 4444);

           envia = new PrintWriter
                        (socketCliente.getOutputStream(), true);
           recibe= new BufferedReader(new InputStreamReader
                                (socketCliente.getInputStream()));
        } catch (IOException error) {
           System.err.println("No se encuentra el equipo: ");
           System.exit(1);
         }
    }

    public String procesaPeticion(String pTrama) throws
                                                    IOException {
        String mensajeRecibido = "";
        StringBuffer lecturaTotal = new StringBuffer();
        try {
           envia.println(pTrama);
           mensajeRecibido = recibe.readLine();
           lecturaTotal.append(mensajeRecibido);

        } catch (IOException e) {
           System.err.println
              ("No existe informacion en el equipo remoto");
           System.exit(1);
         }
        finally {
           envia.close();
           recibe.close();
           socketCliente.close();

        }
        return lecturaTotal.toString();
    }

}
```

Debemos mencionar que estamos haciendo referencia aquí a un Socket Cliente pero debe ser implementado igualmente un Socket Servidor ya que es entre ellos como se entabla la comunicación.

A continuación presentamos un Socket Servidor para que se pueda apreciar el funcionamiento de ambas clases en conjunto, en esencia es lo mismo ya que ambas implementan un PrinWriter y un BuferedReader para entablar una conversación.

La diferencia inicialmente radica en que esta clase implementa un método main que ejecuta esta clase por si sola, no forma parte de la aplicación que estamos desarrollando aunque se una pieza indispensable en su funcionamiento.

El método main o principal, se encarga de preparar el puerto mediante el cual se establecerá la comunicación, por lo tanto debe coincidir este puerto con el de el SocketCliente.

El método run() se encarga de recibir la información proveniente del Socket cliente y es la información que entrará al sistema, se obtendrá una respuesta que será retornada al Socket Cliente por el mismo puerto mandándola mediante la ejecución del método PrinWriter.println().

El constructor de la clase crea un objeto tipo Socket que será quien reciba el flujo de información para que ésta pueda ser interpretada y manipulada como sea requerido.

ServidorSocket.java

```
package conexion;
public class ServidorSocket implements Runnable{

    BufferedReader recibe = null;
    PrintWriter salida = null;
    OutputStream os=null;

    Socket s=null;
    public ServidorSocket(Socket socket){
       s=socket;
    }

    public void run(){
       String mensaje = null;
       try {
          recibe = new BufferedReader(new InputStreamReader
                                        (s.getInputStream()));
          mensaje = recibe.readLine();
          salida=new PrintWriter( s.getOutputStream(), true);
          salida.println(mensaje);
       } catch (IOException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
       }
    }

    public static void main(String[] args) {
       ServerSocket sS=null;
       Socket s1=null;
       try {
              sS= new ServerSocket(4444);
          } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
          }
          try {
              s1=sS.accept();
          } catch (IOException e) {
              // TODO Auto-generated catch block
              e.printStackTrace();
          }

          ServidorSocket ac = new ServidorSocket(s1);
          Thread t= new Thread(ac);
          t.start();
       }
}
```

5.3.7 FormandoResultSet

La información recibida por el Socket cliente a partir del Socket Servidor se espera que sea en una larga cadena de caracteres la cual deberemos desglosar y manipular para así dar forma al ResultSet.

Sabemos que la respuesta en la cadena contiene la información de cada viaje de manera posicional, es decir que obtendremos la información a partir de la cadena sabiendo que a partir de determinada posición encontraremos un dato y en determinada posición concluye dicho dato y para conocer este posicionamiento nos ayudaremos de un Catálogo que estará formado por la clase CatalogoPosicional.

La cadena está dividida en viajes, y estos a su vez en los datos correspondientes a cada viaje. Finalmente este desglose lo realizará la clase ImplementaResultset que veremos en el siguiente tema, lo importante de la clase que aquí mostramos es que de acuerdo a los elementos que explicamos anteriormente permitirán crear un Set(Agrupación) de Resultados (ResultSet).

FormandoResultSet.java

```
package sitio.empresa;

public class FormandoResultSet {
    public    static    ResultSetEmpresa    setResultados(String
tramaViajes) {
        ResultSetEmpresa resultadoViaje = null;
        CatalogoPosicional catPos = new CatalogoPosicional();
        resultadoViaje   =   new   ImplementaResultSet(tramaViajes,
catPos.traeMapaPosicional());
        return resultadoViaje;
    }
}
```

5.3.8 ImplementaResultSet

Como ya se mencionó anteriormente esta clase será utilizada para que implemente la interfase ResultSet para que puedan extraer la información contenida en la cadena o como aquí la llamamos la trama. Por tanto el constructor de esta clase recibe la trama y el catálogo Posicional y establece los métodos a partir de los cuales se extraerá la información proveniente del Socket.

En la clase ViajeDAO hacemos uso de estos métodos tales como getString() y next() que son útiles una vez formado el ResultSet para extraer el dato de la cadena con getString("elemento") y pasándole como parámetro el nombre del elemento que deseamos extraer y así ir formando el primer elemento del ArrayList requerido por el DAO, para pasar al siguiente dato correspondiente, es decir, al siguiente viaje se usa next() y se repite el procedimiento anterior.

ImplementaResultSet.java

```
package sitio.interfases;

import java.sql.SQLException;
import java.util.Map;

import sitio.TO.PosicionTO;

public class ImplementaResultSet implements ResultSetEmpresa {
    private Map posicionMapa;
    private String trama;



    public ImplementaResultSet(String tramaViajes, Map catalogo)
{
        trama = tramaViajes;
    }

    public String getString(String pElemento) {
        try {
            PosicionTO posicionTO = (PosicionTO)
                                    posicionMapa.get(pElemento);
            return trama.substring(posicionTO.getPosicion1(),
                                    posicionTO.getPosicion2());
        } catch (Exception error) {
            System.out.println("No se puede obtener Informacion");
        }
        return null;
    }

    public Object getObjeto() throws SQLException {

        return null;
    }

    public boolean next() {
        if(trama==null)
            return false;
        return true;
    }
}
```

5.3.9 CatálogoPosicional

Este es el catálogo que nos proveerá las coordenadas necesarias para obtener el dato solicitado por ViajeDAO donde se invoca al método getString(“origen”) por ejemplo y la clase ImplementaResultSet consulta el catálogo para encontrar que la clave Origen se encuentra en la posición inicial (0) y se extiende hasta la posición 3, que como podemos apreciar, con ayuda de un TransferObject (PosicionTO) devuelve estas coordenadas a la clase ImplementaResultSet y de esta forma se va formando el ResultSet.

CatalogoPosicional.java

```
package sitio.empresa;

import java.util.HashMap;
import java.util.Map;

import sitio.TO.PosicionTO;

public class CatalogoPosicional {

    public Map traeMapaPosicional() {
       Map mapaPosicion = new HashMap();
       mapaPosicion.put("origen", new PosicionTO(0,3));
       mapaPosicion.put("destino", new PosicionTO(3,6));
       mapaPosicion.put("fecha", new PosicionTO(6,9));
       mapaPosicion.put("hora", new PosicionTO(9,12));
       return mapaPosicion;
    }

}
```

5.3.10 ResultSetEmpresa

Una vez que la respuesta es obtenida desde el Socket todo el flujo de información que hemos ido recorriendo va regresando a la clase de la cual se desprende, el Socket Servidor devuelve la respuesta de la petición al Socket Cliente el cual vio su origen en la clase ConexionEmpresa.java, la respuesta es depositada en un ArrayList pero para que esta información sea devuelta es necesario darle forma de un ResultSet y para ello se ha creado la interfase ResultSetEmpresa para que el DAO pueda manipular el ResultSet de manera habitual.

ResultSetEmpresa.java

```
package sitio.interfases;

import java.sql.SQLException;

public interface ResultSetEmpresa {
    public String getString (String listado);

    public Object getObjeto()throws SQLException;

    public boolean next();

}
```

Gracias a esta interfase el DAO puede generar un ArrayList de TransferObjects lo que permitirá entregar la respuesta requerida por el Proxy el cual podrá entregar un resultado al Servlet y de esta forma será posible direccionar la información hacia una JSP diseñada expresamente para recibir la información y así presentarla al cliente.

5.3.11 JSP - ListadoViajes

Finalmente, una vez procesada la información es necesario proveer al cliente de una respuesta, y ésta vendrá mediante la JSP ListadoViajes donde se generará una sucesión de viajes que se ajusten a los datos solicitados por el cliente en la JSP anterior.

En concreto debemos de apreciar que si hay una razón por la cual manipulamos la información de la manera en que lo hicimos fue a causa de que la información que esperábamos entregar a la JSP así lo requería.

Esta seria la JSP que nos mostrará los resultados de los viajes disponibles para el Origen y Destino solicitados por el cliente en la fecha definida.

ListadoViajes.jsp

```
<%@page import="java.util.ArrayList,
              sitio.TO.ViajeTO,
                "%>

<%
    ArrayList     viajes          =
                     (ArrayList) request.getAttribute("Viajes");
%>
<HTML>
<HEAD>
<TITLE> Listado de Viajes </TITLE>
<LINK REL="Stylesheet" TYPE="text/css" HREF="C:\jakarta-tomcat-
                                    4.1.12\utilerias\estilo.css">

</HEAD>

<BODY>
    <FORM METHOD="POST" NAME="TESIS">
       <TABLE>
          <TR>
             <TD>
                Listado de Viajes existentes para el día:
```

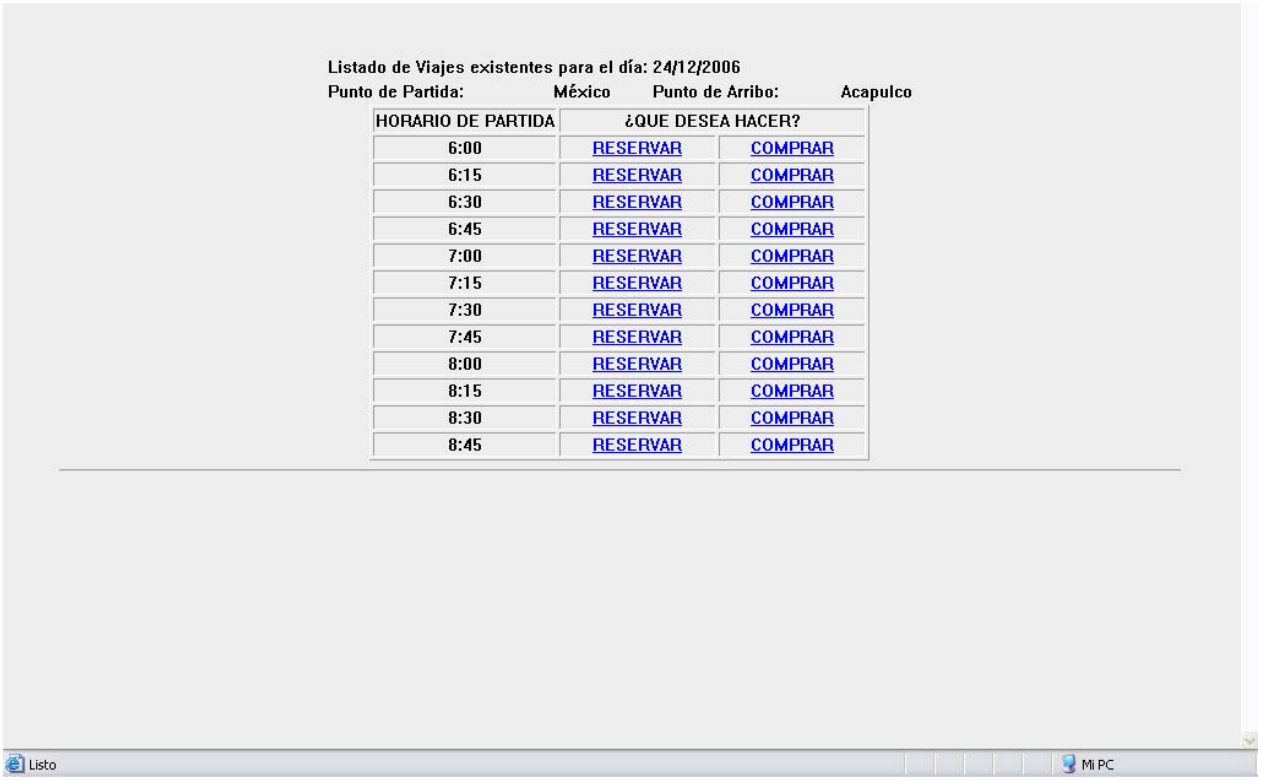
```
          </TD>
          <TD>
                <%= request.getParameter("fecha") %>
          </TD>
       </TR>
       <TR>
          <TD>
             Punto de Partida:
          </TD>
          <TD>
             <%= request.getParameter("origen")) %>
          </TD>
          <TD>
              Punto de Arribo:
          </TD>
         <TD>
                <%= request.getParameter("destino")) %>
          </TD>
     </TR>
</TABLE>
</CENTER>
    <%
       if (viajes != null && viajes.size() > 0) {
    %>

<CENTER>
    <TABLE BORDER="1">
       <TD>
          HORARIO DE PARTIDA
       </TD>
       <TD COLSPAN="2">
          ¿QUÉ DESEA HACER?
       </TD>
          <%
             for (int i = 0 ; i < viajes.size() ; i++) {
                ViajeTO viajeTO = (ViajeTO) viaje.get(i);

          %> <TR>
                <TD>
                   <%= viajeTO.getHora() %>
                 </TD>
                <TD>
                   <A HREF="<%= request.getContextPath()
                               %>/Reservar">RESERVAR</A>
                </TD>
                <TD>
                   <A HREF="<%= request.getContextPath()
                               %>/Comprar">COMPRAR</A>
                </TD>
```

```
                    </TR>
                </TABLE>
            <HR>
            <%
                } else {
            %>
                    <TABLE>
                        <TR>
                            <TD>
                                <CENTER>
                                    <P>
                                        LO SENTIMOS EL VIAJE QUE DESEA
                                         REALIZAR NO ESTA DISPONIBLE
                                    </P>
                                </CENTER>
                            </TD>
                        </TR>
                    </TABLE>
            <%
                }
            %>
        </FORM>
</BODY>
</HTML>
```

Y en el explorador de Internet lucirá de la siguiente manera:



Hasta aquí puede decirse que se cierra un ciclo, el cual esta comprendido desde una JSP hasta otra JSP donde la primera funge como una interfase de petición y la segunda como respuesta hacia el usuario o cliente.

Debe tenerse en cuenta que una JSP en realidad tiene de manera natural un funcionamiento de respuesta y petición, porque otorga al usuario información que extrae y/o procesa de alguna fuente de datos y le permite continuar interactuando con el sitio a través de la misma.

En el caso concreto de la JSP que aquí presentamos, la cual cierra el ciclo del Listado de Viajes extiende la posibilidad de iniciar uno de dos ciclos nuevos, dependiendo de la opción que se elija, RESERVAR o COMPRAR, de acuerdo al horario correspondiente a la elección que realice el cliente.

Nuevamente se desencadenará una serie de procesos que traerán la información correspondiente a la disponibilidad de asientos que hay en el viaje seleccionado, el cliente elige los lugares que va a reservar o a comprar y otra JSP se encargará de conducir los procedimientos correspondientes según sea la selección. Esta sucesión de ciclos en el fondo tienen un comportamiento idéntico a lo que hemos mostrado en el primer ciclo, la única diferencia es que varían las entradas y salidas ya que en el primer caso esperamos un listado de viajes, posteriormente la disponibilidad de asientos, y luego reservación o compra.

El punto es que, si seguimos la lógica de programación planteada podemos crear esta y muchas otras aplicaciones similares debido a que el comportamiento en un ámbito no muy diferente permitirá aplicar la solución planteada, siempre y cuando no se pierdan de vista aquellas particularidades que pudiera contener dicha aplicación.

Bastará entonces con comprender de manera sólida lo que hasta aquí se plantea para que ante un problema afín se conciba de manera general la solución a un problema planteado y a partir de aquí ir incrementando la complejidad en la construcción de aplicaciones según sea necesario para darle a ésta un mejor desempeño y solución a tareas más complicadas.

CONCLUSIONES

A lo largo de esta investigación se trato de abarcar temas que por si solos podrían generar toda una investigación, debido a que son tópicos muy extensos y cuya documentación es muy basta, sin embargo la real intención de la investigación es reunir todos estos elementos en virtud de que, su utilidad práctica requiere que se manejen de manera integrada.

La investigación ha contribuido a localizar aquellos puntos donde el egresado podría tener problemas al presentarse una situación real (práctica), debido a que hilamos los conocimientos adquiridos en nuestra formación académica, con aquellos elementos que podríamos encontrar en el ámbito profesional.

No pretendemos solucionar todas y cada una de las situaciones que el mundo práctico nos puede presentar, ya que cada caso particular presentará nuevas dificultades y retos al desarrollador, sin embargo, resulta ser un buen comienzo conocer el origen de ciertas situaciones a las que nos enfrentaremos para así tener un cimiento firme para seguir impulsándonos hacia adelante.

En cuanto a los temas que aborda esta investigación señalamos que:

El uso de Aplicaciones Web permite la interacción de usuarios o clientes con determinados servicios a través de Internet.

Las aplicaciones Web pueden ser generadas a través de diversos lenguajes y plataformas, pero Java nos ofrece mayores beneficios al no estar predeterminado su uso a determinados sistemas, ya que cuenta con la Máquina Virtual de Java que permite su ejecución en cierta diversidad de sistemas operativos, además de que tiene un respaldo institucional reconocido con empresas como apache y Oracle.

Java esta sustentada en el paradigma de la programación orientada a objetos y por tanto su potencialidad y mantenimiento resultan ideales para proyectos cuyo tamaño resulte amplio, es decir, en proyectos de tipo empresarial que tienden a ser de un gran volumen y su crecimiento requiere modificaciones o adaptaciones recurrentes.

El desarrollo de un sistema puede estar sustentado en diversos patrones de diseño, algunos de los cuales pueden fusionarse para resolver un problema general, algunos de estos patrones coinciden con la metodología que propone la plataforma Java y es por ello que si usamos de manera adecuada la programación orientada a objetos, junto con los patrones de diseño necesarios, y la programación en el lenguaje java, podremos crear a partir de esta fórmula aplicaciones poderosas, flexibles y con un mantenimiento adecuado.

El uso del lenguaje java puede resultar complejo, si embargo nos permite concentrarnos en los problemas relacionados directamente con el funcionamiento de nuestra aplicación, sin tenernos que preocupar en programar componentes y recursos de bajo nivel, ya que estos son elementos que el API de Java ya ha contemplado y por tanto tiene una solución para cada caso.

Las Aplicaciones Web donde existan transacciones de un cliente hacia determinado servicio tienden a ser similares, es por ello que esta investigación puede auxiliar de manera general a resolver problemas de este tipo; aunque no se deben dejar de lado aquellas consideraciones que hacen particular y único a cada caso, ya que será este punto donde tendremos que poner mayor atención y cuidado.

En lo que respecta al conocimiento adquirido a lo largo de este trabajo y como se menciona al comienzo de la investigación, mi caso particular en cuanto a los problemas que enfrenté al mudarme hacia el ámbito práctico, me permitieron visualizar aquellos temas que no tenía dominados e incluso desconocía o no comprendía totalmente para enfocar el trabajo en ese rumbo.

Esta investigación me ha sido útil en muchos sentidos, sobre todo para comprender ciertos casos, que aunque ya habían sido resueltos sobre la marcha, uno no termina de comprender su razón de ser, sino hasta que se ve el panorama completo y se analizan a detalle cada una de sus partes.

APÉNDICE

CASOS DE USO Y DIAGRAMAS

A.1 Casos de Uso

Ejemplo para elaborar documentación referente a la narrativa de casos de uso.

TipoDeDocumento TituloDelDocumento Fecha

CASO DE USO: CONSULTA DE VIAJES

1. Introducción: La consulta de viajes es la operación mediante la cual un cliente puede apreciar la gama de posibilidades que tiene para realizar un recorrido definido en un una fecha determinada. Para que esto sea posible el cliente debe indicar desde donde partirá (origen), hacia donde se dirige (destino) y la fecha en que desea realizar el viaje.
Al resultado obtenido con estos datos se le conoce como consulta de viajes y se mostrará mediante una lista, que desplegará los diferentes tipos de servicio y horarios en que se realizarán los viajes para esa fecha, además cada uno de los viajes estarán precedidos por un par de enlaces que permitirán entrar al sistema de venta o reservaciones.

2. Actores:

Actor	Descripción
Cliente	Proporciona Origen, Destino y Fecha de la Consulta
Sitio	Solicita datos de Consulta a la Empresa y Muestra Resultados de Consulta al Cliente
Empresa	Provee lista de viajes correspondiente a la solicitud del Sitio

3. Detalles:

Meta:	Mostrar al Cliente la gama de viajes que se ajusten a la solicitud de consulta del cliente.
Precondiciones:	Que los servicios que provee la empresa estén disponibles, así como los servidores que intervienen en el proceso.
Poscondiciones (Éxito):	El sitio debe permitir al cliente realizar la consulta de viajes, así como desplegar la información solicitada de manera precisa y actual. El sitio permitirá a partir de la consulta de viajes ingresar al servicio de venta o reservación si el cliente lo requiere.
Poscondiciones (Fracaso):	El sitio deberá notificar al cliente por medio de mensajes en caso de que se presente algún error generado por la consulta, sea por que no existe el viaje, una fecha incorrecta (anterior a la fecha actual), o que el origen y/o destino no sean definidos o iguales. Si el servicio que provee la empresa no responde o no esta disponible también se le debe indicar al cliente la situación.
Disparador:	El actor que genera el funcionamiento de este caso de uso será el cliente a través del sitio al momento presionar el botón Consulta Viajes. Donde previamente debieron haber sido definidos el origen, destino y fecha de la consulta.

CASO DE USO: REGISTRO DE CLIENTE

1. Introducción:

El registro de cliente solo es necesario en el caso de que se desee hacer una reservación o venta y éste consistirá únicamente en pedir el nombre completo del cliente y su dirección de correo electrónico.

2. Actores:

Actor	Descripción
Cliente	Introducirá en una casilla su Nombre completo y en otra su dirección de correo electrónico y procederá a registrarse.
Sitio	Solicita los datos del cliente, lo registrará introduciendo esa información en la base de datos e iniciará la sesión del cliente en el sitio.

3. Detalles:

Meta:	Crear un registro con la información del cliente en la base de datos e iniciar su sesión en el sitio. En el caso de que los datos introducidos ya existan en la base de datos no se creará un nuevo registro solo se actualizará la información en el registro existente.
Precondiciones:	Que el cliente introduzca sus datos correctamente, no aceptando que las casillas queden en blanco y que la casilla de dirección de correo electrónico siga la estructura estándar (nombreusuario@dominio.com).
Poscondiciones (Éxito):	El sitio, tras haber introducido sus datos el cliente: - Permita registrar al cliente. - Cree el registro en la base de datos. - Inicie una sesión en el Sitio. - Presente la disponibilidad de asientos del viaje seleccionado.
Poscondiciones (Fracaso):	El sitio debe indicar al cliente si éste ha introducido información no valida, ya que en este caso no se generará registro alguno.
Disparador:	El formulario de registro de cliente aparece justamente después de que el cliente elige hacer una reservación o compra por primea vez (si ya se ha registrado se inicia una sesión y si la sesión esta iniciada ya no es necesario registrarse). El registro ocurre en el momento en que se dé clic en el botón registrar, tras haber introducido sus datos correctamente en el formulario.

CASO DE USO: DISPONIBILIDAD DE ASIENTOS

1. Introducción: La disponibilidad de asientos se refiere a la relación de asientos ocupados y asientos disponibles que hay en un viaje específico, de tal suerte que, cuando un usuario elija un viaje pueda visualizar mediante la página Web, un esquema del autobús el cual le permita elegir los asientos que desee reservar o comprar.

2. Actores:

Actor	Descripción
Sitio	Se encarga de dibujar el **diagrama** de disponibilidad de asientos correspondiente al viaje elegido y permitirá al cliente realizar la elección de asientos disponibles mediante el mismo **diagrama**.
Empresa	Entregará la disponibilidad de asientos del viaje al sitio para mostrar el **diagrama**.

3. Detalles:

Meta:	Mostrar el diagrama completo del autobús mostrando la distribución de los asientos y si estos están ocupados o disponibles.
Precondiciones:	Debe escoger un viaje y elegir la opción de reservar o comprar. El servicio que la empresa provee para esta situación este disponible
Poscondiciones (Éxito):	El diagrama que se muestra esté completo y con datos actuales,
Poscondiciones (Fracaso):	El servicio que la empresa provee para esta situación no esté disponible.
Disparador:	Al momento de elegir venta o reservación en un viaje en la sección consulta de viajes

CASO DE USO: SELECCIÓN DE ASIENTOS

1. Introducción: La disponibilidad de asientos estará representada mediante un diagrama que esquematice al autobús y la distribución de sus asientos (el diagrama diferenciará con una imagen los asientos ocupados, con otra los disponibles y los seleccionados con otra), de esta forma el cliente podrá visualizar y elegir aquellos lugares que desee reservar o comprar, para que esto ocurra, el cliente podrá hacer clic en las imágenes que representen lugares disponibles (al momento de hacer clic cambiará la imagen por la de asiento seleccionado), repetirá esta acción tantas veces como sea necesario para marcar todos aquellos lugares que solicite (haciendo doble clic sobre el asiento seleccionado, éste regresará a su estado de disponible).

2. Actores:

Actor	Descripción
Cliente	El cliente hará clic en las imágenes de los asientos disponibles que desee reservar o comprar.
Sitio	Mostrará una página Web con la información del viaje seleccionado, condiciones de apego a políticas de reservación o venta (según sea el caso), el diagrama del autobús.

3. Detalles:

Meta:	Seleccionar (apartar) los asientos que el cliente desee reservar o comprar
Precondiciones:	Si las políticas de la empresa correspondientes a las reservaciones y ventas lo permiten se podrá realizar la reservación o compra.
Poscondiciones (Éxito):	El sitio permitirá seleccionar los asientos disponibles al Cliente
Poscondiciones (Fracaso):	Si las reservaciones destinadas para ese viaje se han agotado.
Disparador:	Al hacer clic en los asientos disponibles

CASO DE USO: DESBLOQUEO DE ASIENTOS

1. Introducción: El desbloqueo de asientos ocurre como una medida correctiva de la disponibilidad de asientos, la cual ocurrirá en el caso de que una reservación o una compra, no sea concluida de manera exitosa, por ejemplo, al momento de detectar 3 solicitudes de autorización rechazadas consecutivamente por el servicio del banco, por tanto, el sitio procederá a desbloquear los asientos que se intentaban comprar en esta sesión.
Esto ocurre por razones seguridad ya que las políticas del banco establecen que no debe ocurrir que una solicitud de autorización sea rechazada más de 3 veces.

2. Actores:

Actor	Descripción
Cliente	No concluye el trámite de reservación o venta.
Sitio	No permite continuar con el proceso de reservación o venta.
Banco	Rechaza una solicitud de crédito 3 veces consecutivas.
Empresa	Desbloquea Asiento

3. Detalles:

Meta:	Restablecer disponibilidad de asientos bloqueados.
Precondiciones:	Estado de asiento bloqueado, La solicitud de autorización de crédito sea rechazada. Que el proceso de compra o reservación no sea concluido.
Poscondiciones (Éxito):	Un asiento bloqueado regrese a su estado de disponibilidad después de una transacción fallida.
Poscondiciones (Fracaso):	El servicio de disponibilidad de asientos de la Empresa no esté disponible.
Disparador:	Procesos de venta o reservación fallidos, inconclusos o rechazados.

CASO DE USO: RESERVACIÓN EN LÍNEA

1. Introducción: Una vez que el cliente ha seleccionado sus asientos deberá hacer clic en el enlace reservar, lo cual desencadenará una petición de reservación a la empresa, si esta petición es aceptada por la empresa ésta devolverá un número de reservación que el sitio mostrará al cliente junto con los datos del viaje, los asientos reservados, el costo de la reservación, además enviará un correo electrónico a la dirección del cliente y finalmente se le pedirá al cliente que imprima el comprobante mediante el cual podrá realizar la compra de sus boletos en taquilla. El comprobante contendrá un mensaje informativo útil para el cliente.

2. Actores:

Actor	Descripción
Cliente	Hará clic en el enlace de reservar e imprimirá el comprobante de su reservación.
Sitio	Enviará la petición de reservación a la empresa con los datos del viaje y los asientos seleccionados y mostrará la respuesta al cliente.
Empresa	Recibe la petición, autoriza o rechaza la reservación de acuerdo con la disponibilidad de asientos actualizada y las políticas restrictivas al respecto. Si la petición es autorizada devolverá un número de reservación.

3. Detalles:

Meta:	Ejecutar la reservación de asientos y devolver un comprobante imprimible al cliente.
Precondiciones:	Que sea seleccionado al menos un asiento y además que no se rebase el número de reservaciones definidas por las políticas de reservación.
Poscondiciones (Éxito):	La empresa devolverá un número de reservación, el cual será mostrado por el sitio junto con los demás datos de la reservación para que el cliente imprima esta información
Poscondiciones (Fracaso):	La disponibilidad de asientos actual puede haber sufrido cambios durante el proceso de selección de asientos, de tal suerte que si la empresa encuentra alguna coincidencia entre un asiento seleccionado y uno ocupado no se realizará la reservación y la empresa devolverá la disponibilidad de asientos actualizada.
Disparador:	Cuando el cliente hace clic en el enlace Reservar.

CASO DE USO: VENTA EN LÍNEA

1. Introducción: El proceso de Venta ocurre a partir de que el formulario de pago con tarjeta es llenado por el cliente y el sitio después de realizar las validaciones necesarias no encuentra error, por tanto se procede a solicitar una petición de crédito al Banco y devuelve una respuesta al sitio para que esta sea presentada al cliente.

2. Actores:

Actor	Descripción
Cliente	Hace clic en el enlace de comprar y recibe respuesta donde se le notifica en caso de algún error, si el resultado es exitoso la empresa le devolverá un número de operación con el que podrá recoger sus boletos en taquilla
Sitio	Envía petición de crédito al Banco y Despliega la respuesta al Cliente con la posible falla (si es rechazada) o con el número de operación (si es aceptada).
Banco	Recibe la petición de crédito y la autoriza o la rechaza
Empresa	Espera autorización de crédito para vender los boletos y entrega al cliente el número de operación a través del sitio.

3. Detalles:

Meta:	Obtener autorización de crédito del banco para vender los boletos
Precondiciones:	Que el cliente tenga crédito con el banco. Que el servicio del banco este disponible.
Poscondiciones (Éxito):	Al autorizar el banco el crédito, la empresa realiza la venta de boletos.
Poscondiciones (Fracaso):	Si no se autoriza el crédito los boletos no son vendidos el cliente deberá asegurarse que los datos introducidos son correctos o usar otra tarjeta y reintentar la compra.
Disparador:	El cliente hace clic en el enlace de Comprar en el formulario de pago con tarjeta de crédito.

CASO DE USO: INTERFASE DE COMUNICACIÓN

1. Introducción: La comunicación directa con la base de datos del sistema de la empresa no es posible por razones de seguridad y por que está montado en una plataforma no estándar, la cual se maneja de una manera distinta, por tanto, la manera en que las operaciones se enviarán desde el sitio hacia el sistema de la empresa será mediante un componente conocido como **interfase de comunicación cliente,** el cual se encargará de transportar la información proveniente hacia el servidor donde residirá la contraparte denominada **interfase de comunicación servidor,** la cual transformará esta información en una cadena de caracteres (trama), susceptible de ser procesada por el sistema de la empresa. Para cada tipo de operación la empresa establece una estructura posicional que debe cumplir la trama, (estas estructuras podrán ser consultadas en el Anexo de esta investigación). De manera inversa la empresa entregará una respuesta al sitio en la misma forma (una trama), y la interfase de comunicación deberá decodificar dicha trama para transformarla en información útil para el sitio.
En el caso de una transacción donde tenga que intervenir el Banco, la interfase de comunicación trabajará conjuntamente con el componente de Conexión Banco para la comunicación con este Actor.

2. Actores:

Actor	Descripción
Sitio	El sitio define una interfase para comunicarse con el banco y con la empresa
Empresa	La empresa recibe una petición de información por parte del sitio, la resuelve y devuelve el resultado al sitio.
Banco	Recibirá una solicitud de crédito por medio del componente de Conexión Banco y la interfase de comunicación, la resolverá y devolverá por este medio un resultado al sitio.

3. Detalles:

Meta:	Permitir la comunicación entre el sitio y la empresa o entre el sitio y el banco
Precondiciones:	Los servicios de la empresa o del banco (según sea necesario) estén en funcionamiento y que los datos sean proporcionados exactamente como solicitan dichos servicios
Poscondiciones (Éxito):	La empresa responde a una solicitud del sitio Que el banco autorice la transacción que se solicita.
Poscondiciones (Fracaso):	El servicio de la empresa no responde o no funciona. El servicio del banco rechace la solicitud de crédito o esté fuera de funcionamiento.
Disparador:	En el momento en que el sitio solicita una conexión con el banco o la empresa

.CASO DE USO: CONEXION BANCO

1. Introducción: Debido a que la interacción directa entre el sitio y el banco no es posible por razones de seguridad del propio banco, éste establece una forma específica en la que se debe interactuar con él. Por tanto, será necesario crear un componente de software capaz de comunicarse con el servicio que el banco proporcionará.
La manera en que se debe comunicar el sitio con el Banco es mediante largas cadenas de caracteres las cuales estarán compuestas por secuencias de letras y números que forman claves que indican la información que se desea comunicar al banco o la respuesta que éste comunica.
A partir de la solicitud del cliente, el sitio construirá la larga cadena de caracteres con los datos contenidos en la solicitud del cliente (atributos), y la enviará al servicio del banco como un parámetro, mediante la invocación de un método contenido en una DLL que permitirá la interacción con el banco. De igual forma, el banco entregará al sitio una respuesta en una cadena de caracteres la cual deberá ser interpretada para detectar si la solicitud fue aceptada o no, además de desplegar al cliente la respuesta a su solicitud.
De manera paralela y en el caso de que sea aceptada la solicitud, el sitio debe comunicarse con la entidad conocida como empresa, para que se realice el bloqueo de asientos y la venta de dichos boletos.

2. Actores:

Actor	Descripción
Sitio	Formará la cadena de caracteres que será enviada como argumento del método que invocará para interactuar con el banco. Recibirá una cadena de caracteres como respuesta la cual interpretará para proporcionarle al cliente el resultado de su operación
Banco	Al momento de recibir la cadena de caracteres, este actor se encarga de realizar una serie de procesos internos donde le informará al sitio si es aceptada o no la autorización mediante una constante tipo cadena de 80 caracteres.
Cliente	Visualiza el resultado de la operación, mediante la página y en caso de existir algún error, se le estará informando para que lo corrija y reenvié la solicitud.

3. Detalles:

Meta:	Que el banco autorice la solicitud de crédito para realizar la venta.
Precondiciones:	Que el servicio de banco este en funcionamiento. Que los datos proporcionados al banco sean correctos Que el cliente tenga viabilidad de crédito.

Poscondiciones (Éxito):	El sitio se comunicará con el sistema de la empresa para realizar el bloqueo de asientos, concluir la venta de los boletos, finalmente le será entregada la respuesta al cliente.
Poscondiciones (Fracaso):	El cliente no tiene crédito disponible, o introdujo incorrectamente sus datos. Alguno de los servicios, el banco o la empresa no estén disponibles.
Disparador:	El proceso de venta en línea invocará el método en el momento en que se complete el armado de la cadena de caracteres, que será enviada como argumento de dicho método.

CASO DE USO: PAGO CON TARJETA DE CRÉDITO

1. Introducción: Luego de seleccionar los asientos a comprar y hacer clic en el enlace de continuar, el sitio mostrará una página con los datos correspondientes al viaje, los asientos seleccionados, el costo de los boletos en detalle y además una sección donde el cliente deberá de introducir la información correspondiente a la tarjeta de crédito con la cual realizará el pago.

2. Actores:

Actor	Descripción
Cliente	Introduce los datos de la tarjeta de crédito con la que efectuará el pago.
Sitio	Presenta la información del viaje, el costo de cada uno de los boletos, sus descuentos, y recopila la información de la tarjeta de crédito del cliente.

3. Detalles:

Meta:	Obtener información de la tarjeta de crédito para armar la solicitud de aprobación de crédito al banco
Precondiciones:	Que la información presentada sea la que el cliente espera. Que los datos de la Tarjeta sean introducidos correctamente de acuerdo a las validaciones previstas.
Poscondiciones (Éxito):	Que el Cliente introduzca sus datos y los de la tarjeta de manera correcta.
Poscondiciones (Fracaso):	Si las validaciones que el sitio realiza encuentran alguna anomalía no se podrá continuar con la venta.
Disparador:	Al momento de hacer clic en el enlace de continuar en la página abocada a la selección de asientos

CASO DE USO: CORREO ELECTRÓNICO

1. Introducción: Tanto en el caso de la reservación como en el de la venta, al tiempo de ser éstas aprobadas, el sitio enviará un correo electrónico al cliente con las instrucciones y datos correspondientes a la operación realizada.

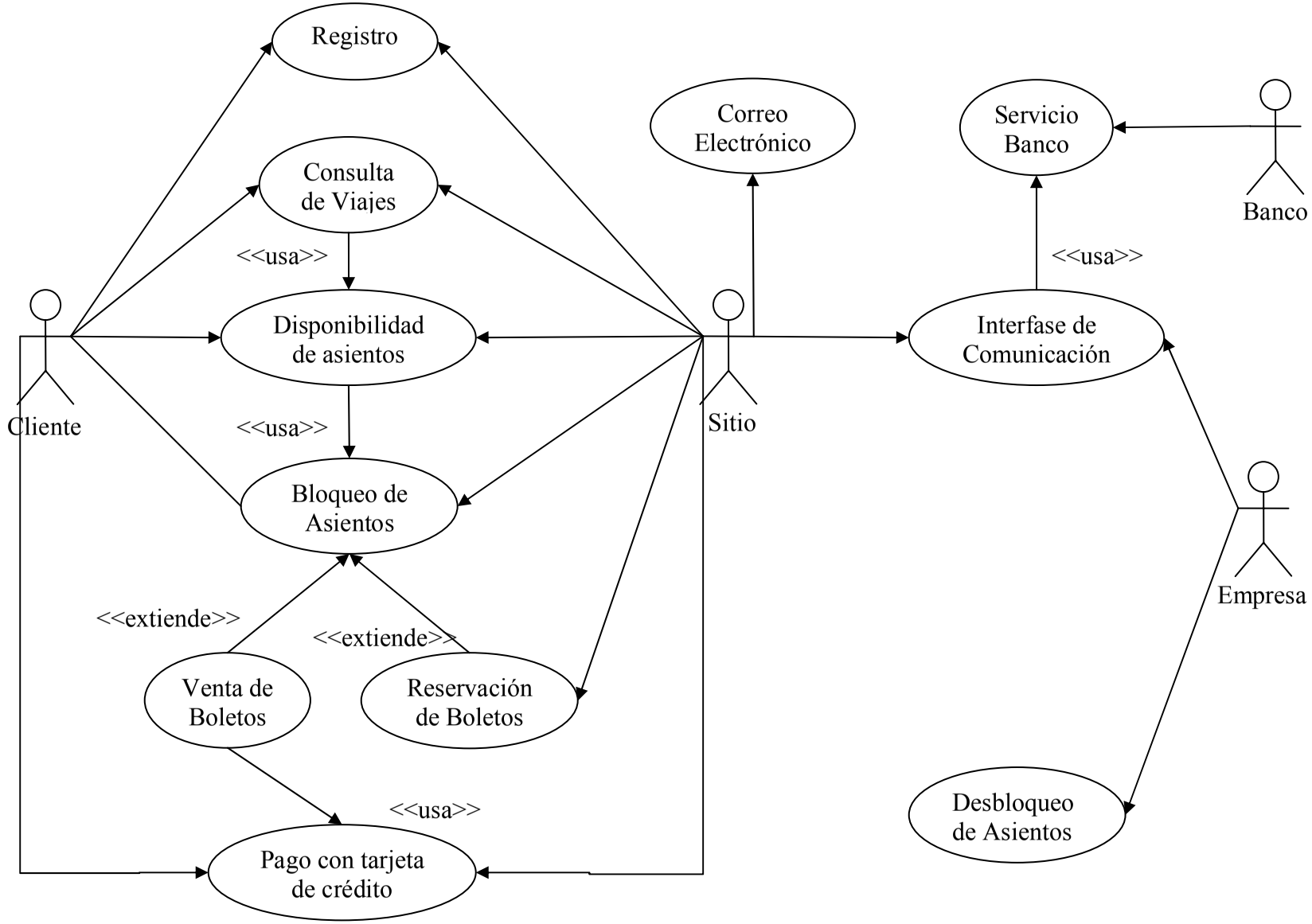
2. Actores:

Actor	Descripción
Empresa	La empresa aprueba la solicitud de reservación o compra.
Sitio	Envía un correo electrónico al cliente en el momento en que la reservación o la compra son aceptadas.

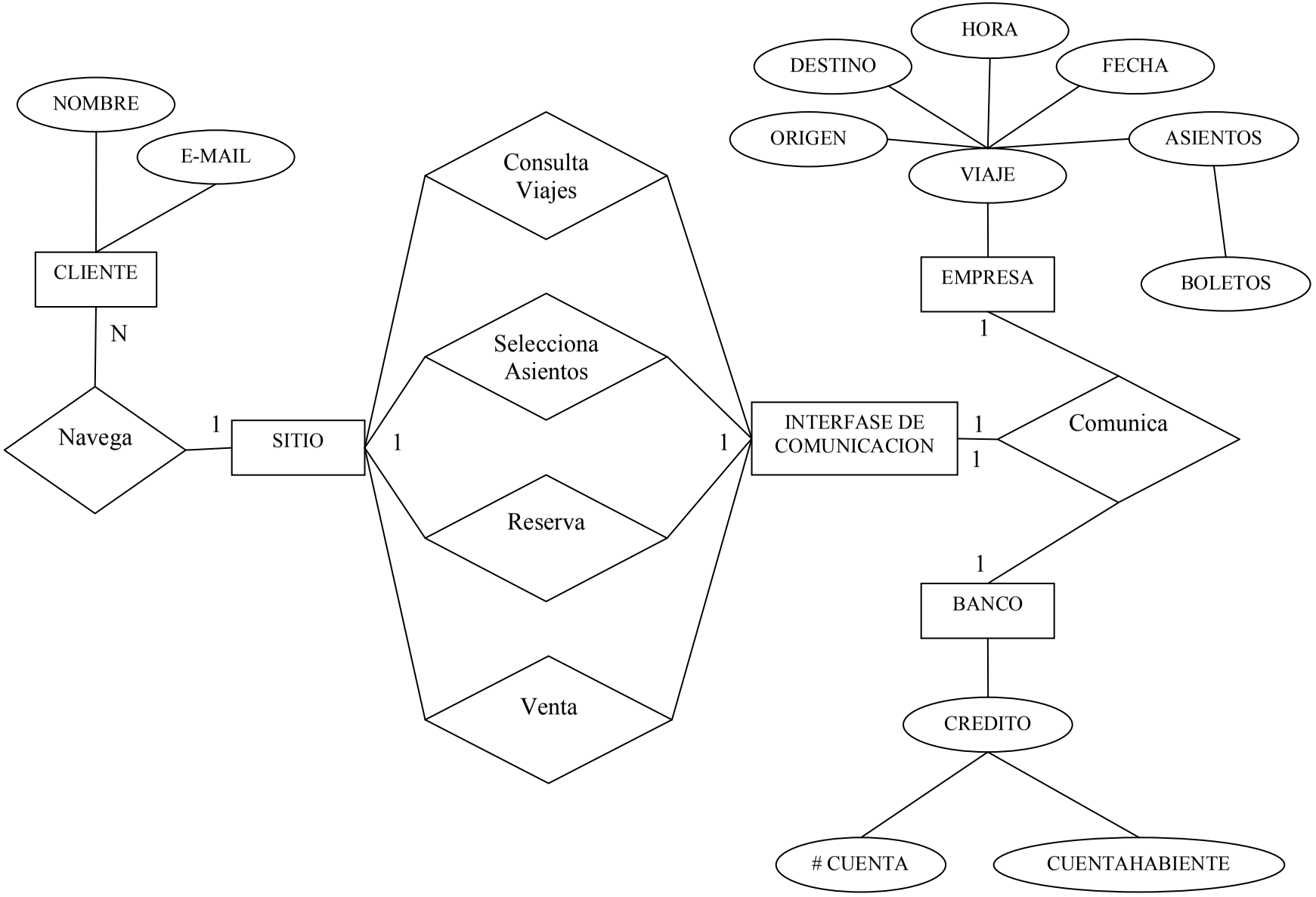
3. Detalles:

Meta:	Envió de correo electrónico al cliente con datos correspondientes a la reservación o compra según sea el caso.
Precondiciones:	Que la reservación o la compra sean aprobados
Poscondiciones (Éxito):	Entrega de correo electrónico en el buzón del cliente.
Poscondiciones (Fracaso):	Si el cliente no proporciona una dirección de correo real o la introduce de manera incompleta o incorrecta.
Disparador:	La aprobación de la reservación o de la compra.

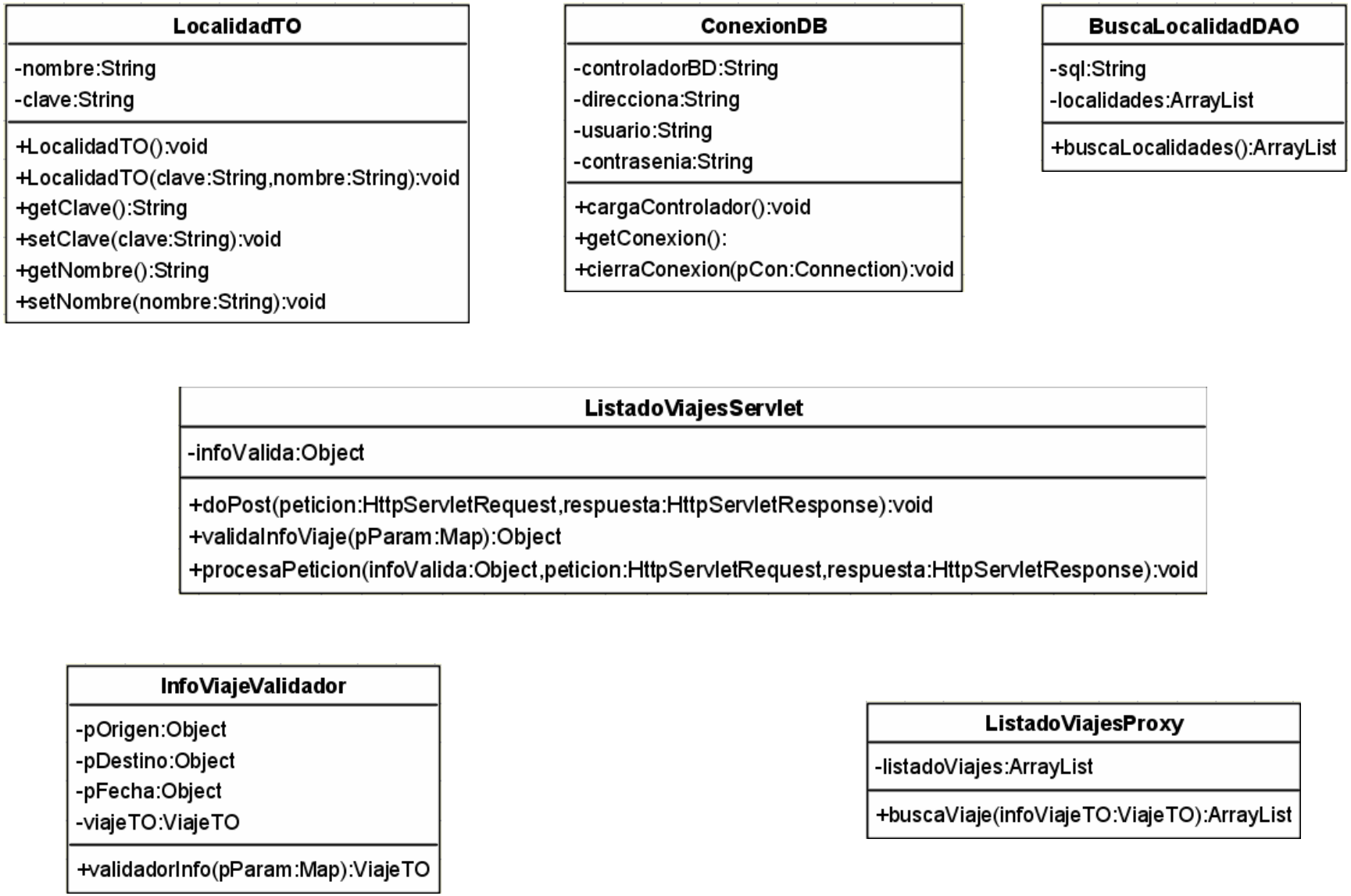
A.2 Diagrama de Casos de Uso

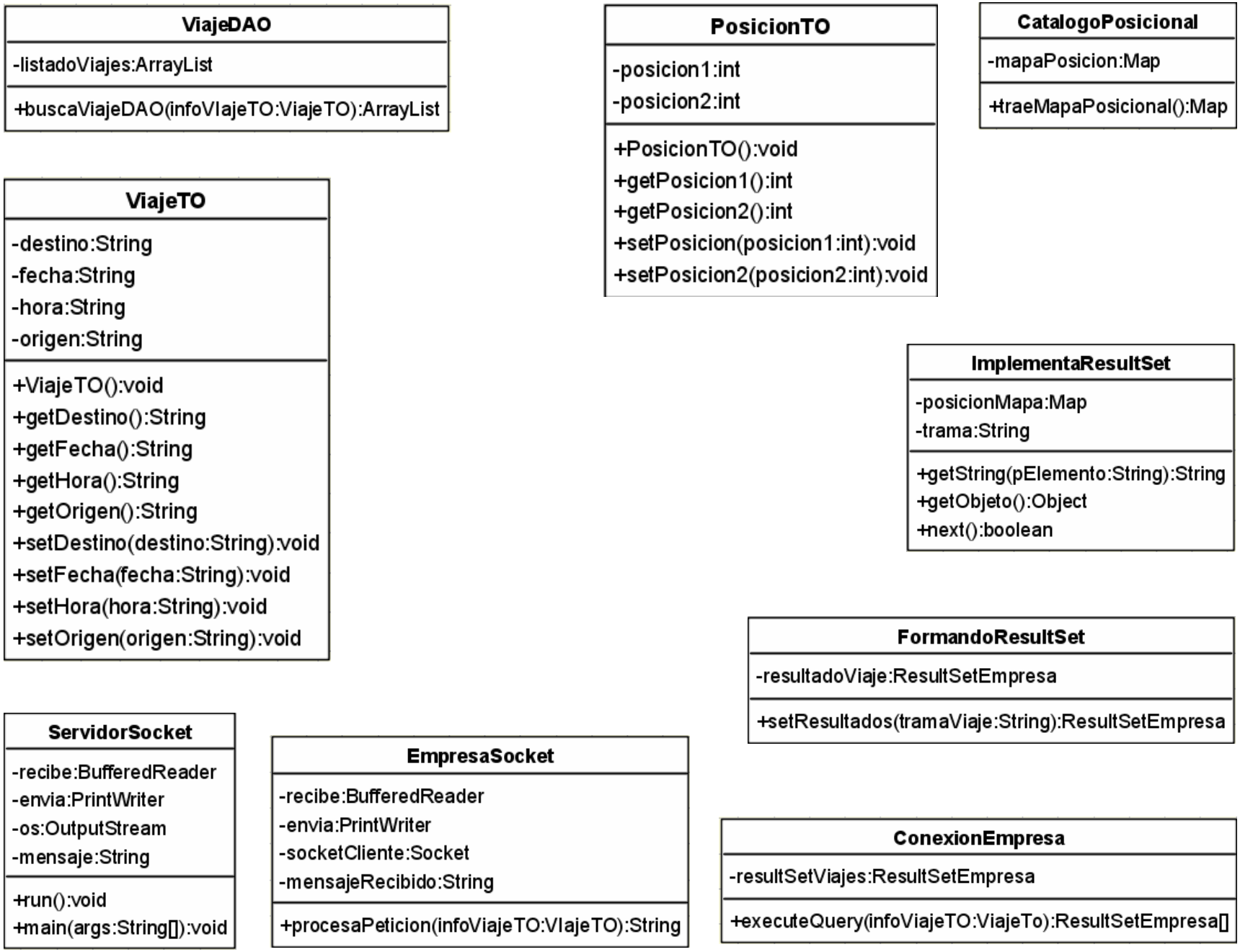


A.3 Diagrama Entidad - Relación



A.4 Diagrama de Clases




ViajeDAO
-listadoViajes:ArrayList
+buscaViajeDAO(infoVIajeTO:ViajeTO):ArrayList
PosicionTO
-posicion1:int
-posicion2:int
+PosicionTO():void
+getPosicion1():int
+getPosicion2():int
+setPosicion(posicion1:int):void
+setPosicion2(posicion2:int):void
CatalogoPosicional
-mapaPosicion:Map
+traeMapaPosicional():Map
ViajeTO
-destino:String
-fecha:String
-hora:String
-origen:String
+ViajeTO():void
+getDestino():String
+getFecha():String
+getHora():String
+getOrigen():String
+setDestino(destino:String):void
+setFecha(fecha:String):void
+setHora(hora:String):void
+setOrigen(origen:String):void
ImplementaResultSet
-posicionMapa:Map
-trama:String
+getString(pElemento:String):String
+getObjeto():Object
+next():boolean
FormandoResultSet
-resultadoViaje:ResultSetEmpresa
+setResultados(tramaViaje:String):ResultSetEmpresa
ServidorSocket
-recibe:BufferedReader
-envia:PrintWriter
-os:OutputStream
-mensaje:String
+run():void
+main(args:String[]):void
EmpresaSocket
-recibe:BufferedReader
-envia:PrintWriter
-socketCliente:Socket
-mensajeRecibido:String
+procesaPeticion(infoViajeTO:VIajeTO):String
ConexionEmpresa
-resultSetViajes:ResultSetEmpresa
+executeQuery(infoViajeTO:ViajeTo):ResultSetEmpresa[]

A.5 Diccionario de Datos

Tabla Localidad

CAMPO	TIPO	NULL?	LONGITUD	DESCRIPCIÓN
loc_clave	Caracter	Not Null	4	Clave única de identificación de la localidad
loc_nombre	Caracter	Not Null	25	Nombre de la localidad a la que se viaja

Tabla Cliente

CAMPO	TIPO	NULL?	LONGITUD	DESCRIPCIÓN
cli_clave	Caracter	Not Null	4	Clave única de identificación del cliente
cli_nombre	Caracter	Not Null	50	Nombre del Cliente
cli_mail	Carácter	Not Null	50	Dirección de Correo Electrónico del Cliente

Estas son las únicas tablas utilizadas a partir de la base de datos referida. Todos aquellos datos a los que el sistema se refiere en cuanto a viajes, disponibilidad de asientos, etc., son elementos propios de la empresa y por tanto, esos datos son proporcionados por ésta con el fin de regir la estructura de entrada y salida de la información para darle la estructura deseada a la trama o bien para que la trama sea interpretada.

GLOSARIO

A

Acoplamiento: El término hace referencia al grado de interdependencia entre módulos (capas). El cual puede ser considerado, dentro de una escala del más fuerte (el menos deseable) al más débil (el más deseable).

Aplicación: Es una pieza de software que permite explotar la potencialidad de la computadora proveyendo al usuario de la realización de una tarea específica.

Aplicación Web: Es un elemento de software que es ejecutado a través de un explorador de Internet y que además de explotar la potencialidad que ofrece la computadora, puede usar los recursos que Internet o algún otro tipo de red provea.

API: Interfaz De Programación De Aplicaciones (Application Programming Interface). Es un conjunto de especificaciones disponibles para que el programador haga uso de rutinas o funciones que se encargan de situaciones comunes en el desarrollo de software.

Arreglo: Es una estructura de programación utilizada para contener información de manera secuenciada en una, dos o "n" dimensiones.

Atributos: Son características que definen el comportamiento y forma de un elemento o control.

B

Base De Datos: Es un conjunto de registros almacenados en una computadora, a través de una organización sistemática realizada por un sistema de administración que hace posible consultar, modificar y agregar información.

C

Caso De Uso: Es una técnica para capturar requerimientos funcionales de sistemas.

Clase: Es una colección de métodos y variables los cuales son usados de manera conjunta para crear Objetos

Cliente: Es un entidad informática que permite acceder de manera remota a otro equipo de cómputo a través de alguna red.

Clustering: Hace referencia al enlace de servidores individuales física y programáticamente y a la coordinación de la comunicación entre ellos, de modo que puedan realizar tareas comunes.

Compilar: Es la acción de procesar código generado por el programador para convertirlo en código ejecutable.

Componente: Es un objeto que se ajusta a determinada especificación, es un elemento de software que resuelve una problemática determinada.

Constructor: Es el elemento de una clase cuyo objetivo es preparar al objeto para ser creado,

Contenedor: Es una clase encargada de administrar otros objetos, es decir, que es un almacén de elementos programados que provee recursos de bajo nivel, los cuales pueden ser utilizados por otras clases para resolver determinada problemática.

Controlador: Es un programa que gestiona el funcionamiento de un dispositivo o en el caso de las bases de datos, establece un enlace entre una aplicación y el sistema de administración de la base de datos.

E

En Línea (On Line): Se utiliza esta expresión para definir cuándo se realiza una actividad haciendo uso de una red computacional.

Excepción: En Java una excepción ocurre cuando un error se presenta durante la ejecución de la aplicación.

Explorador De Internet O Web Browser: Es una aplicación que permite navegar en Internet haciendo posible visualizar páginas e información que reside en la red.

Extranet es una Intranet extendida, es una red virtual constituida por dos o más intranets, que juntas permiten a Internet transportar información entre sus terminales

F

Failover Ocurre cuando en una estructura donde residen varios servidores, alguno de estos dejara de funcionar, existe el proceso que automáticamente conmuta la carga de trabajo a otro servidor para proporcionar un servicio continuo.

H

Hardware Se refiere a los componentes físicos que integran un equipo de cómputo

Hipertexto Se refiere a cualquier texto disponible en la red que contenga enlaces con otros documentos. Utilizar el hipertexto es una manera de presentar información en la cual texto, sonido, imágenes y acciones están enlazados entre sí, de manera que se pueda pasar de una a otra en el orden que se desee.

HTML (Hypertext Mark-Up Language – Lenguaje de Marcas de Hipertexto): Es el lenguaje utilizado para generar páginas Web que a las postre serán accedidas mediante un explorador de Internet bajo el protocolo HTTP.

HTTP (HyperText Transfer Protocol): Protocolo para la transferencia de Hipertexto, es una convención mediante la cual se define que la transferencia de información se realizará a través de Hipertexto.

I

Internet: Es la interconexión de más de 100,000 redes de computadoras en todo el mundo. La cual permite que aproximadamente 50 millones de usuarios de computadoras intercambien información y correo electrónico.

Intranet: Es una red local que utiliza los protocolos de Internet para comunicarse internamente en una institución.

J

Java: Es un lenguaje de programación desarrollado por la empresa Sun MicroSystems que propone la idea de programación: escribe una vez y ejecútalo en cualquier lugar, (Write Once Run Everywere).

Javabean: Es un componente de Java que permite la transferencia de datos, almacenándolos y entregándolos mediante los métodos conocidos como getters y setters.

Javaserver Pages: Es un componente de Java que se caracteriza por su ejecución en el servidor y que permite otorgarle dinamismo a una página HTML, ya que son lenguajes que se pueden combinar para funcionar de manera conjunta.

Java Servlet: Es una clase que se ejecuta del lado del servidor, acepta peticiones de clientes y genera respuestas dinámicamente mediante un protocolo de petición/respuesta.

JDBC: El API JDBC es un API para la conectividad con sistemas de bases de datos relacionales.

M

Mainframe: Así se les conoce a las supercomputadoras, es decir, equipos potentes y costosos que están abocados a realizar tareas delicadas de procesamiento como pueden ser transacciones bancarias

N

Nativo: Se refiere a una instrucción que es implementada de manera específica y directa con el procesador sin hacer uso de algún elemento externo.

P

Parámetro: Es una variable cuyo valor permite modificar el comportamiento de la rutina a la que representa mediante el valor que le sea asignado

Portal: Es la página de inicio que permite el acceso a las distintas secciones de un sitio Web

Q

Query: Palabra inglesa que se refiere a una consulta que se hace a una base de datos para obtener información específica.

R

Red: Conjunto de equipos de cómputo conectados entre si con el objetivo de compartir información.

Resultset: Es una colección de datos obtenidos a partir de la información que arroja una base de datos a partir de una consulta (Query)

S

Servidor De Aplicaciones (Application Server): Es una tecnología básica que proporciona la infraestructura que brindará los servicios y componentes necesarios, para implementar un sistema basado en determinada plataforma.

Sitio: Colección de páginas Web asignadas a una dirección Web única.

SOAP: Es un protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML, cuya utilización se da en servicios Web.

Sistema Operativo: Es un conjunto de programas que permiten hacer uso de los recursos básicos con los que cuenta un equipo de cómputo, ya que estos sistemas necesitan un entorno básico para su funcionamiento.

Servidor: Este término se utiliza para hacer referencia al dispositivo físico, pero también para hacer referencia al Software que lo hace funcionar; de tal suerte que, puede causar confusión al usar el término fuera de contexto. Una computadora cualquiera, que cuenta con este software instalado, también se le considera como servidor. Por ello resulta técnicamente mejor decir que un Servidor es un conjunto de Hardware y Software que responde a los requerimientos de un cliente.

Socket: Es un componente de dos vías que se comunica con su contraparte (otro socket) ubicados en computadoras distintas con la finalidad de intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

Software: Aquellos elementos lógicos que integran un sistema computacional.

Stream: Este término se refiere al flujo de información.

W

Workflow: Se refiere al flujo de trabajo, es decir, aquellos aspectos de operación que genera una actividad de trabajo.

X

XML: eXtensible Markup Language (lenguaje de marcas extensible) es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas, para compartir la información de una manera segura, fiable y fácil.

BIBLIOGRAFÍA

CÁRDENAS, Alfonso Araujo, *Sistemas Distribuidos Concepto y Características.* (Articulo), 2004.

DÍAZ, Moisés Daniel, *Diseño de aplicaciones Internet usando los Patrones de diseño J2EE.*

FISTEUS, Jesús Arias, *Modelado de procesos de negocio. Aplicación en entornos móviles.* (Tesis Doctoral), Madrid 2002.

FRIAS, Jeann José, *Estudio y desarrollo de la seguridad en el comercio electrónico entre dos entidades productivas a través de Internet.* (Tesis de Maestría), México, 2000.

FROUFE, Quintas Agustin, *JavaServer Pages: Manual de usuario y tutorial.* Madrid, RA-MA Editorial, 2002.

HALL, Marty, *Servlets Y JavaServer Pages. Guía Práctica.* México, Pearson Educación, 2001.

MOLPECERES, Alberto Touris y PÉREZ, Martín Mariñán, *Arquitectura empresarial y software libre, J2EE.* España, javaHispano, 2002.

OELKERS, Dotty Boen, *Comercio Electrónico, Serie Business.* México, Internacional Thomson Editores, 2003.

PEÑALOZA, Romero Ernesto, *Fundamentos de Programación*. México, UNAM, 2001

VALOR, Josep. et. al. *Aprovechando las oportunidades del B2B.* (Ponencia), 2003.