Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

" SISTEMA DE RESERVACIONES "

TESIS
PROFESIONAL

PARA OBTENER EL TÍTULO DE:
LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
ANA ELYN HERNÁNDEZ GONZÁLEZ

ASESOR:
M.C. PEDRO BELLO LÓPEZ

PUEBLA, PUE. **MAYO 2008**

AGRADECIMIENTOS

Han sido muchas las personas que me han apoyado a lo largo de mi vida y de mis estudios, sin ellas no hubiera logrado cerrar este capítulo de mi vida y comenzar otro, del que espero continúen siendo mi mayor motivación.

A mis padres.

Esta tesis va dedicada principalmente a ustedes, que han sido un ejemplo a seguir durante mi vida y me han enseñado que la perseverancia y el esfuerzo son el camino para lograr objetivos, por creer en mi, por la educación y sobre todo por el gran apoyo y amor.

A mis hermanos.

Gracias por sus consejos, apoyo, críticas, respeto y ánimos para seguir adelante.

A mis Amigos.

Por su amistad, apoyo, comprensión y consejos que me han dado.

Mtro. Pedro Bello López.

Un agradecimiento a mi asesor por su compresión, tiempo, comentarios, aportaciones y dedicación en la supervisión de este trabajo.

A mis Maestros de la Facultad.

Por sus enseñanzas.

INDICE

INTRODUCCIÓN.

La era de la información nos ha alcanzado y el mundo gira en torno a la computación, en los corporativos, empresas e instituciones, la automatización de procesos se ha convertido en una tarea necesaria para mejorar la productividad, reducir costos, agilizar los tiempos e incrementar la calidad entre muchos otros beneficios.

Con la introducción del Internet y del Web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso y uso de información desde cualquier parte del mundo. Con los avances en tecnología cada vez se demandan aplicaciones más rápidas, ligeras y robustas que permitan ser usadas sin importar el lugar u horario.

Estas aplicaciones estarán diseñadas para interactuar con bases de datos con el fin de organizar y distribuir información.

Las aplicaciones Web ofrecen grandes ventajas que pueden ser aprovechadas por muchas organizaciones, sobre todo ahora que la globalización es una realidad. Entre las ventajas que se pueden mencionar están:

- ✓ **No requieren instalación**, pues usan tecnología Web, lo cual nos permite el aprovechamiento de todas las características del Internet.
- ✓ **Son fáciles de usar** (no requieren conocimientos avanzados de computación).
- ✓ **Alta disponibilidad**, ya que puede realizar consultas en cualquier parte del mundo donde tenga acceso a Internet y a cualquier hora.
- ✓ **Reducción de trabajo Administrativo**
- ✓ **Transacciones más rápidas y precisas**

El propósito de este proyecto de tesis es automatizar los principales procesos administrativos en base al diseño y construcción de un sistema de reservaciones para un salón de eventos sociales, con el propósito de centralizar la información, cuidar la integridad de los datos y tener impacto en agilizar la productividad, los tiempos de respuesta, automatizar la reservación del evento, ofreciendo un servicio de calidad y eficiencia hacia este proceso.

Para poder conseguir lo antes mencionado se utilizaron una serie de herramientas tanto de ingeniería de software, como de base de datos, entre las que se encuentran el modelo relacional y el modelo entidad-relación.

Para el desarrollo del sistema de reservaciones utilizamos el modelo de cascada que inicia con el análisis y diseño, finaliza con la implementación y pruebas de desempeño.

Este trabajo está estructurado en cinco capítulos en los que se describe el entorno de desarrollo del sistema, el análisis de requerimientos, diseño del sistema, la implementación y las pruebas de desempeño.

- ✓ En el capítulo I se presentan las herramientas necesarias para el desarrollo del sistema, entre ellas: análisis y diseño orientado a objetos, definiciones de JSP y Servlets de Java, ingeniería de software, UML, Bases de datos relacionales, normalización; todo esto conforma el marco teórico de este documento.

- ✓ En el capítulo II se muestra el análisis del proyecto a través del planteamiento del problema, los antecedentes, los beneficios y las fortalezas, la metodología utilizada, los requerimientos funcionales y no funcionales, la identificación de usuarios, el modelado de casos de uso y finalizando con el diseño conceptual de la base de datos.

- ✓ En el capítulo III se describe el diseño de la base de datos, haciendo uso de diagramas de estado, diagramas de comunicación, diagramas de interacción y diagramas de clases. Se definen las entidades, los atributos de cada entidad, su dominio, y las relaciones entre las entidades, y con esto queda definido el diseño lógico de la base de datos.

- ✓ En el capítulo IV queda definida la implementación y pruebas necesarias para el funcionamiento adecuado del sistema, se realizan las interfaces y traducción del diseño a un lenguaje de programación, que posteriormente quedara plasmado en la aplicación de este trabajo.

- ✓ En el capitulo V presenta las pruebas de desempeño del sistema y los resultados obtenidos, lo que permite verificar que se hayan cumplido los requerimientos del sistema.

Finalmente en las conclusiones, se habla de los puntos más importantes que se realizaron durante el desarrollo de la tesis, mencionando también las perspectivas que se tienen del mismo y el trabajo a futuro que se podría realizar en base al trabajo presentado aquí.

1.5. Análisis y Diseño Orientado a Objetos (ADOO).

La tendencia actual del software demanda la construcción de sistemas más grandes y más complejos, el paradigma de programación orientada a objetos presenta una buena propuesta a los desarrolladores para llevar a cabo esto. Para crear sistemas orientados a objetos de buena calidad, es necesario realizar un Análisis y un Diseño previos, y éstos se realizan mucho mejor si se hace uso de una notación robusta y se sigue una metodología de trabajo.

En este capítulo se describe en qué consiste el Análisis y Diseño Orientado a Objetos, así como UML y el proceso unificado, que son respectivamente, una notación y una metodología auxiliares en dicho proceso.

El problema fundamental que se debe asumir en el desarrollo de software es convertir el mundo real en un programa informático. El desarrollo de software orientado a objetos implica la creación de modelos del mundo real y la construcción de programas informáticos basados en esos modelos.

Un problema de programación orientada a objetos describe normalmente como un conjunto de especificaciones (detalles que constituyen el problema real). Las especificaciones son parte de lo que se denomina **Análisis Orientado a Objetos (AOO)**, que responde a la pregunta ¿Qué hace? Durante la fase de análisis se piensa en las especificaciones en términos intuitivos y con independencia del lenguaje y de la máquina. La etapa crítica de esta actividad es deducir los tipos de objetos del mundo real que están implicados y obtener los atributos de estos objetos determinando su comportamiento e interacciones [Joyanes 98]. Por lo tanto, el análisis se centra en la investigación del problema, no en la manera de definir una solución.

La siguiente fase del proceso de desarrollo de software es el **Diseño Orientado a Objetos (DOO)**, que responde a la pregunta ¿Cómo lo hace? Durante esta fase se comienza a crear un modelo de computadora basado en el análisis que realice la tarea específica concreta. En esta etapa se piensa en objetos del mundo real que pueden ser representados como objetos del mundo informático. Se deben especificar los objetos con mayor precisión, especificando en detalle lo que los objetos conocen y lo que pueden hacer, y describe con prudencia sus interacciones [Joyanes 98]. Es decir, el diseño pone de relieve una solución lógica: cómo el sistema cumple con los requerimientos.

La diferencia entre AOO y DOO no es clara y es difícil definir la transición entre ambas etapas. De hecho estas fases no representan un proceso estricto de dos etapas y a veces se funden en una sola.

La esencia del **Análisis y Diseño Orientado a Objetos** consiste en situar el dominio de un problema y su solución lógica dentro de la perspectiva de los objetos [Larman 99].

1.5.1.UML

El UML (Unified Modeling Language – Lenguaje Unificado de Modelado) es un intento para desarrollar un meta-modelo que unifique semánticas y del cual pueda ser construida una notación común [Hunt 00]. Es decir, es una notación que incorpora lo mejor de otras notaciones de modelado, cuyo objetivo es ser un vocabulario común para construir sistemas orientados a objetos.

El UML está hecho de un número de modelos que juntos describen el sistema a diseñar. Cada modelo comprende uno o más diagramas con documentación y descripciones de soporte. Cada modelo es creado con la intención de ser una descripción del sistema a ser diseñado desde una perspectiva particular. Cada diagrama puede ser parte de más de un modelo, o diferentes partes del mismo diagrama pueden ser partes de diferentes modelos.

Los diagramas principales que comprende UML son los siguientes (ver Figura 1.1) [Hunt 00]:

- Diagramas de casos de uso: presentan las interacciones entre usuarios (humanos u objetos) y el sistema. Resaltan la funcionalidad primaria del sistema.

- Diagramas de clases: presentan la estructura estática del sistema. Son el centro de la notación UML y del diseño orientado a objetos.

- Diagramas de objetos: usan notación que es casi idéntica a los diagramas de clases, pero presentan los objetos y sus relaciones en un punto particular en el tiempo. Los diagramas de objetos no son tan importantes como los diagramas de clases pero pueden ser muy útiles.

- Diagramas de actividad: describen el flujo de actividades o tareas, típicamente dentro de una operación. Son algo como pseudocódigo gráfico.

- Diagramas de secuencias: muestran la secuencia de mensajes enviados entre objetos que se encuentran colaborando en una tarea particular. Resaltan el flujo de control entre objetos.

- Diagramas de colaboración: muestran también, la secuencia de mensajes enviados entre objetos que se encuentran colaborando en una tarea en particular. Pero resaltan las relaciones entre los objetos que están colaborando.

- Diagramas de estado: ilustran los estados en los que un objeto puede estar y las transiciones que mueven al objeto entre estados.

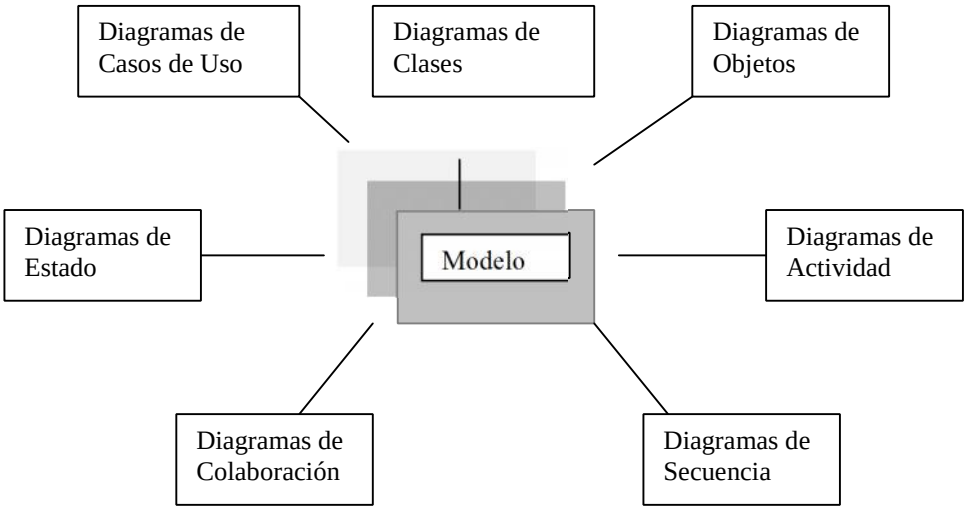


Figura 1.1 Relaciones entre diagramas y modelos según [Hunt 00].

Un diagrama representa una vista particular en el modelo. Idealmente el sistema completo podría ser descrito en un diagrama sencillo y fácil de comprender. Sin embargo sólo los sistemas más sencillos pueden lograr dicho objetivo. En lugar de eso, un modelo de un sistema de software sólo será eso: un modelo de la situación real. Los modelos ocultan algunos de los detalles de la situación real. En ese sentido, un modelo UML presenta una vista particular de un sistema, abstrayendo u ocultando detalles particulares.

Incluso un modelo sencillo es una entidad compleja y es un tanto difícil presentar una vista significante dentro de un solo diagrama. Un modelo puede ser visto en diferentes formas. Cada diagrama de UML es dicha vista.

Uno de los aspectos clave del UML es que cada diagrama debería ser consistente con cualquier otro diagrama que represente la misma información. Esto es, si una operación es mencionada en dos diagramas ésta debería tener el mismo nombre, con el mismo tipo de retorno y con los mismos parámetros, en cada uno.

Cuando un diseño es documentado usando UML, se crean múltiples modelos. Cada modelo captura un aspecto diferente del diseño emergente. Estos aspectos son documentados en términos de diagramas y notas adicionales. El elemento clave de cada modelo es el aspecto visual del diseño; sin embargo este aspecto visual es argumentado por descripciones textuales y especificaciones.

El UML es por tanto un lenguaje para visualizar, especificar, describir y documentar un sistema de software [Hunt 00].

Sin embargo UML no es un método de diseño, es puramente una notación para documentar un diseño. Una notación no es suficiente, se requiere de un método que indique como aplicarla.

Conceptualmente UML puede ser usado con cualquier método de diseño orientado a objetos apropiado. El método del Proceso Unificado es introducido en los siguientes párrafos.

1.5.2. El Proceso Unificado

Todos los métodos de Análisis y Diseño Orientado a Objetos más significativos que han surgido son concentrados en la arquitectura, incrementales e iterativos. No adoptan el modelo tradicional de cascada para el desarrollo de software, en lugar de eso adoptan una aproximación que va más de acuerdo al modelo de espiral de Bohem [Hunt 00]. Y el Proceso Unificado es uno de ellos.

El Proceso Unificado es un sistema de diseño que guía las tareas, la gente y los productos en el proceso del diseño. Es un sistema porque provee las entradas y las salidas de cada actividad, pero no restringe indicando como cada actividad debe ser realizada. Diferentes actividades pueden ser usadas en diferentes situaciones, algunas eliminándose, otras siendo reemplazadas o argumentadas.

El objetivo principal del Proceso Unificado es definir [Hunt 00]:

- Quién hace qué
- Cuándo lo hacen
- Cómo alcanzar cierta meta
- Las entradas y salidas de cada actividad

Es entonces un proceso de ingeniería que comprende un número de diferentes elementos jerárquicos, como se muestra en la Figura 1.2

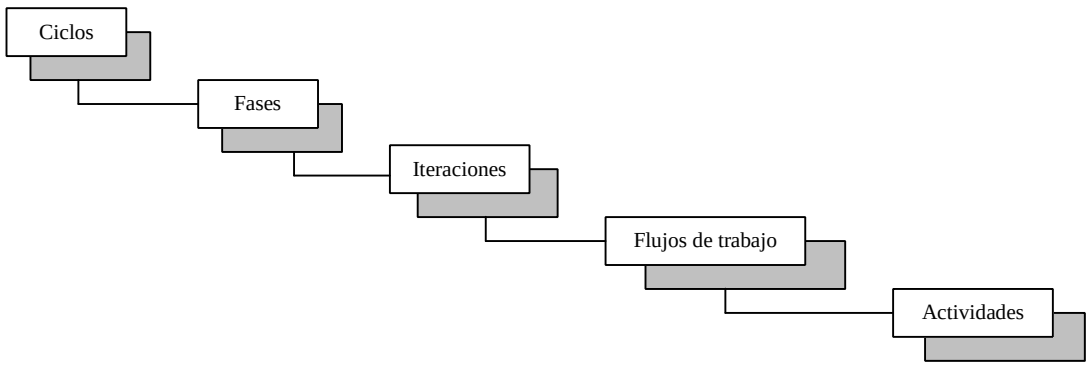


Figura 1.2 Bloques de construcción claves en el proceso Unificado según [Hunt 00].

El Proceso Unificado comprende actividades de bajo nivel (tales como encontrar clases), que son combinadas en flujos de trabajo. Estos flujos de trabajo están organizados en iteraciones. Cada iteración identifica algunos aspectos del sistema a ser considerados. Las iteraciones por si mismas son organizadas en fases. Las fases se enfocan en diferentes aspectos del proceso de diseño, por ejemplo requerimientos, análisis, diseño e implementación. A su vez las fases pueden ser agrupadas en ciclos. Los ciclos se enfocan en la generación de versiones sucesivas de un sistema.

Hay cuatro elementos clave en la filosofía detrás del Proceso Unificado [Hunt 00]:

- Es iterativo e incremental
- Es dirigido por casos de uso
- Es centrado en la arquitectura
- Hace reconocimiento de riesgos

1.5.3.Dirigido por casos de uso

El Proceso Unificado es también dirigido por casos de uso, ya que estos ayudan a identificar los requerimientos primarios del sistema.

En el Proceso Unificado los casos de uso son usados para asegurar que el diseño que se está desarrollando es siempre relevante a lo que requiere el usuario. Así los casos de uso actúan como un hilo conductor a través de todo el proceso de desarrollo, como se muestra en la Figura 1.3

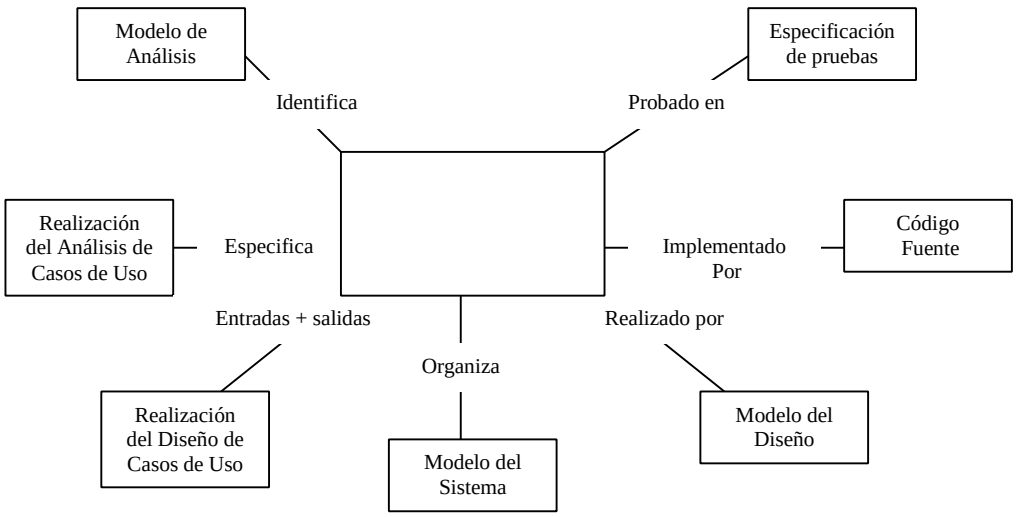


Figura 1.3 El papel de los casos de uso según [Hunt 00].

Por ejemplo al comienzo de la fase de diseño una de las dos entradas principales es el modelo de casos de uso. Entonces, explícitamente dentro del modelo del diseño, hay relaciones de casos de uso que ilustran cómo cada caso de uso es soportado por el diseño.

Para resumir el papel de los caso de uso, ellos:

- Identifican los usuarios del sistema y sus requerimientos
- Auxilian en la creación y validación de la arquitectura del sistema
- Ayudan a producir la definición de casos de prueba
- Dirigen la planeación de iteraciones
- Dirigen la creación de documentación de usuario
- Dirigen la organización del sistema
- Sincronizan el contenido de diferentes modelos
- Dan seguimiento a través de los modelos

1.6. Java Servlets

Los Servlets son componentes del servidor. Estos componentes pueden ser ejecutados en cualquier plataforma o en cualquier servidor debido a la tecnología Java que se usa para implementarlos. Los Servlets incrementan la funcionalidad de una aplicación Web. Se cargan de forma dinámica por el entorno de ejecución Java del servidor cuando se necesitan. Cuando se recibe una petición del cliente, el contenedor/servidor Web inicia el servlet requerido. El Servlet procesa la petición del cliente y envía la respuesta de vuelta al contenedor/servidor, que es enrutada al cliente [Allamaraju 00].

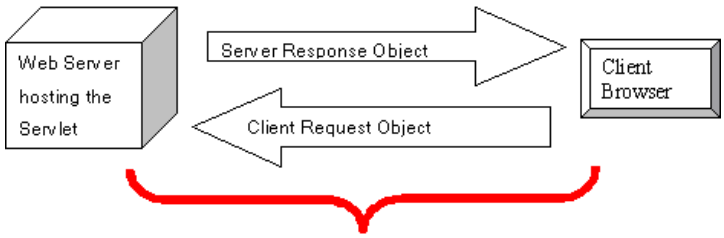


Figura 1.4: Modelo de respuesta a peticiones http.

La interacción cliente/servidor basada en Web usa el protocolo HTTP. EL protocolo HTTP es un protocolo sin estados basado en un modelo de petición y respuesta con un número pequeño y finito de métodos de petición como GET, POST, HEAD, OPTIONS, PUT, TRACE, DELETE, CONNECT, etc.. La respuesta contiene el estado de la respuesta y meta-información describiendo dicha respuesta. La mayor parte de las aplicaciones Web basadas en servlets se construyen en el marco de trabajo del modelo petición/respuesta HTTP (Figura 1.4).

1.6.1. Requerimientos de Ejecución de los Servlets Java

- Los Servlets de Java requieren algún conocimiento previo de Java y HTML.
- Las aplicaciones Web y las páginas Web que requieran iniciación de servlets deben ejecutarse en servidores Web con contenedores Web integrados, como el servidor web iPlanet [iPlanet] o un contenedor solitario de servlets como el Tomcat [Tomcat].
- El API Servlet (que tratemos de entender en breve) suele estar mayoritariamente integrado en el servidor/contenedor web. El contenedor/servidor logra esto último implementando la especificación Java Servlet 2.1 o 2.2 [Sun].

1.6.2. Ciclo de Vida de Servlets

Todos los servlets siguen el modelo del ciclo de vida figura 1.5. El contenedor web tiene la responsabilidad de crear una instancia del servlet y de invocar al método *init* (1). Si un cliente ha enviado una petición al contenedor web, entonces, esa petición se pasa al método *servicio* del servlet (2), y se envía una respuesta de vuelta al contendor web (3). Finalmente, cuando el servlet haya finalizado su propósito, el contenedor web invoca al método *destroy* (4).

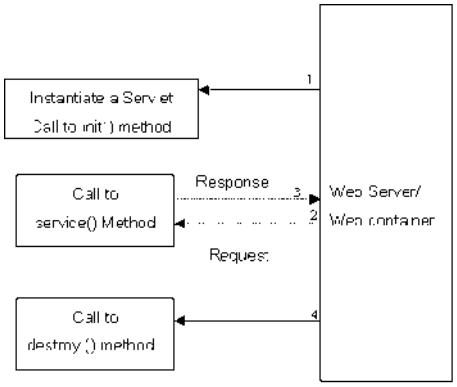


Figura 1.5: Ciclo de Vida del Servlet

Además de los métodos que heredan de la clase GenericServlet, la clase HTTPServlet tiene métodos adicionales para cada uno de los métodos de respuesta HTTP tratados antes.

- `doDelete (HttpServletRequest, HttpServletResponse)`
- `doGet (HttpServletRequest, HttpServletResponse)`
- `doOptions (HttpServletRequest, HttpServletResponse)`
- `doPost (HttpServletRequest, HttpServletResponse)`
- `doPut (HttpServletRequest, HttpServletResponse`
- `doTrace (HttpServletRequest, HttpServletResponse)`

Por ejemplo, la clase Servlet tiene que redefinir `doGet` o `doPost` (o ambos métodos, dependiendo de si los datos serán enviados por un GET o por métodos de petición POST HTTP). Estos métodos toman dos parámetros: un objeto `HttpServletRequest` y un objeto `HttpServletResponse`. Estos dos objetos dan acceso total a toda la información sobre la petición del cliente, y ayudan a controlar la salida enviada al cliente como respuesta a dicha petición. Los métodos `doGet` y `doPost` son invocados por el método *service*. El método *service* puede usarse directamente al redefinirlo.

1.6.3. Ventajas de los Servlets sobre los CGI

Los servlets añaden comportamiento dinámico a los servidores. La API de programación de los servlets hace muy fácil la escritura de servicios complejos para aplicaciones basadas en web, sin tener que centrarse en los detalles de bajo nivel de los protocolos HTTP, formatos de petición, y cabeceras. Puesto que los servlets tienen un marco de trabajo similar al Java, son independientes de la plataforma tanto en el cliente como en el servidor. Además, los servlets pueden ser enlazados con diferentes bases de datos, como Oracle, Servidores SQL, etc... y también con varios servidores web.

Los Servlets de java son extensiones directas del servidor web. Simplemente, son objetos Java que se cargan de forma dinámica por el Entorno de Ejecución del Java (JRE) cuando se necesitan. Cuando un proceso se inicia en un servidor, necesita la asignación de varios recursos. El cambio entre procesos implica también mucha sobrecarga debido al cambio de contexto al tener que grabar toda la información de un proceso para volver más tarde a él.

A un proceso se le puede llamar hilo pesado debido a que inicia un proceso completo, con una enorme cantidad de sobrecarga en términos de tiempo, memoria, etc.. Por el contrario, por cada petición a un servlet, se crea un hilo ligero para manejarla. Un hilo ligero es un proceso hijo, que es controlado por el proceso padre (en este caso, el servidor). En tal escenario, el cambio de contexto se hace muy fácil, y los hilos pueden pasar fácilmente de activos a inactivos, o a espera. Esto mejora sustancialmente el rendimiento de los servlets sobre los scripts CGI.

Para afinar el rendimiento, con los servlets es posible "cachear" la información común (como las consultas a bases de datos) en la memoria, para evitar los costosos e innecesarios accesos a la base de datos. No existe ninguna forma sencilla de implementar esto en un CGI. Los Servlets también tienen un manejo muy potente de las excepciones.

Sin embargo, existen determinados casos en los que la creación de servlets no sería un requerimiento. Podría ser una buena inversión sólo si el sitio web recibe muchas visitas; para sitios web menos visitados, se pueden realizar las mismas tareas con scripts CGI [Marty Hall 01].

1.7. Introducción a JSP

JSP significa Java Server Pages, que traducido quiere decir Páginas de Servidor Java y es por tanto una tecnología orientada a crear páginas web con programación Java, es decir, permite mezclar HTML estándar y estático con un contenido dinámico; etiquetas especiales (órdenes) y trozos de código Java (scriptlets o secuencias de órdenes) o contenido dinámico generado por servlets. Éste contenido dinámico se encuentra encerrado en etiquetas especiales, muchas de las cuales inician con <% y finalizan con %>. El motor de las páginas JSP está basado en los servlets de Java.

El concepto inherente en un archivo JSP es permitir ver un servlet Java como una página HTML. Esto elimina la desagradable lista de sentencias *print()* que normalmente lleva todo el código java del servlet. Es decir, una página JSP trata de permitir que se pueda incluir código java dentro de una página HTML normal.

Una página JSP es preprocesada a un archivo .java, que luego es compilado para generar un archivo .class. Esta es la innovación que proporciona la tecnología *Javaserver Pages* y que la diferencia de otras semejantes, por ejemplo, una página *Active Server Page (ASP)*, se compila en memoria, no en un archivo separado [Froufe 02].

Cuando en un servidor web se recibe la petición de una página JSP, éste lanza el motor JSP, que se ejecuta en el mismo proceso que ese servidor web. El motor JSP comprueba si la página es nueva o ha cambiado, en cuyo caso realiza el *proceso de traslación* de la página y luego compila el resultado. El proceso de traslación es la parte principal del funcionamiento de la tecnología JSP, y consiste en la conversión de la página JSP en un servlet java. Este servlet es compilado mediante el compilador java estándar y ejecutado utilizado el API estándar de java.

El proceso de traslación es el que ralentiza la ejecución de la página JSP; sin embargo, una vez que la página JSP ha sido convertida a servlet y compilada, su ejecución es tan rápida como si su origen hubiese sido un servlet normal. La figura 1.6 muestra los componentes involucrados en el procesamiento de la petición HTTP realizada por el cliente de una página JSP:

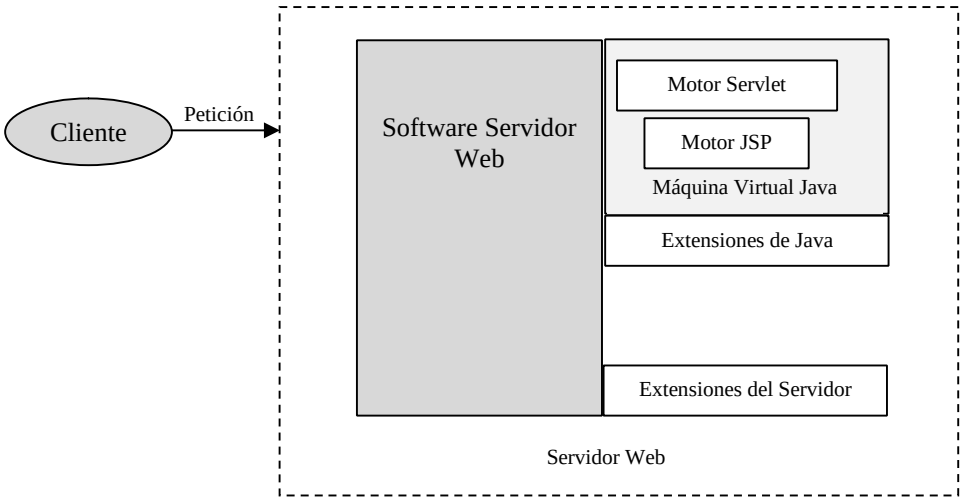


Figura 1.6 Petición Cliente

Como muestra la figura anterior todo el tratamiento de la petición HTTP que se hace en el servidor web hasta que se devuelve la respuesta al cliente, se resume en cuatro pasos:

1. El motor JSP analiza la página solicitada y crea un fichero de código fuente java correspondiente a servlet.
2. El servlet generado se compila para obtener el archivo .class, que pasa el control del motor servlet, que lo ejecuta del mismo modo que si se tratase de cualquier otro servlet.
3. El motor servlet carga la clase del servlet generado para ejecutarlo.
4. El servlet se ejecuta y devuelve su respuesta al solicitante.

El proceso es mucho más eficiente de lo que puede deducirse. El análisis de la página y la traslación a servlet solamente ocurre una vez, la primera ocasión en que se realiza la solicitud de la página, o cuando se modifica ésta. La carga de la clase del motor servlet también ocurre una vez desde la última vez que se haya arrancado el motor servlet. Después de esto, el servlet ya está disponible durante toda la vida de la máquina virtual java. Incluso para mejorar la eficiencia del último paso, hay algunos servidores web que proporcionan un mecanismo de *caché de páginas* que reduce el coste de ejecutar la petición y mejorar el rendimiento de la aplicación; es decir, mediante este mecanismo, la generación de la respuesta puede realizarse una única vez, o bien solamente cuando su contenido dinámico haya variado.

1.7.1.Arquitecturas Jsp.

Al momento de describir las arquitecturas que se pueden utilizar en el desarrollo de aplicaciones web, dos son las estructuras más frecuentes.

La primera estructura es la que toma el modelo cliente-servidor como paradigma, lo que hace que todas las peticiones realizadas por el cliente lleguen directamente a la página JSP; modelo page-centric.

La segunda estructura es la que utiliza n-capas, que consiste en la utilización de este modelo en donde un servlet o página JSP actúa como controlador de la aplicación, delegando peticiones y respuestas a páginas JSP.

1.7.1.1. Modelo Cliente-Servidor.

Ésta es la arquitectura que ha sido más utilizada, y que consiste en una o más aplicaciones ejecutándose sobre máquinas cliente y conectadas a un servidor para poder trabajar.

En este modelo se permite el acceso directo desde la página JSP a cualquier recurso para satisfacer la petición del cliente. La página JSP es la que procesa la petición y envía la respuesta al cliente. En este caso, una página JSP solamente se diferencia de un servlet en que permite escribir un código más claro, y en que se puede agrupar el código Java en JavaBeans separándolo de lo que es la generación de la presentación de la respuesta al cliente.

La ventaja de este modelo es que resulta muy fácil de programar y el contenido dinámico de una página se genera rápidamente, en base a la petición del cliente y al estado de los recursos y además,

a la hora de realizar modificaciones, solamente hay que abrir un archivo. La figura 1.7 muestra un diagrama con las partes más relevantes de este modelo de arquitectura.

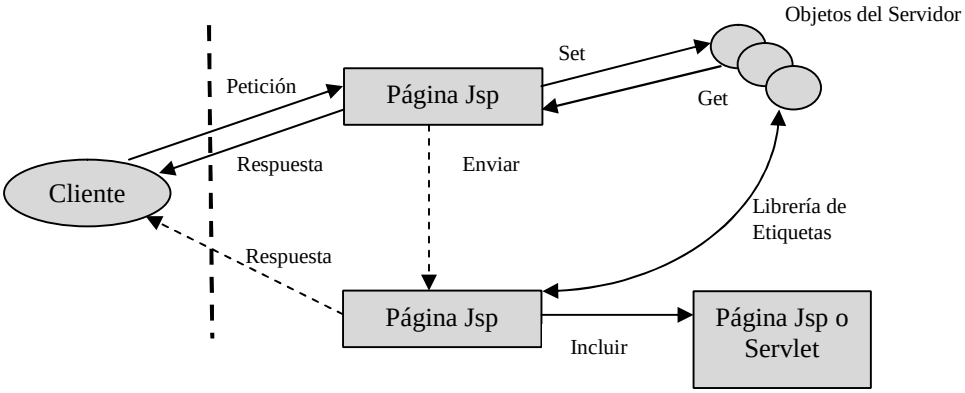


Figura 1.7: Modelo Cliente-Servidor.

Sin embargo, la principal desventaja de este modelo es que no es escalable; un gran número de peticiones simultáneas puede requerir el uso de muchos recursos del servidor, lo cual hará disminuir el rendimiento de la aplicación web [Froufe 02].

1.7.1.2. Modelo n-capas.

En este modelo hay un servlet o página JSP que actúa como controlador de las peticiones, pasándolas a JavaBeans, servlets o páginas JSP específicas. Al interrelacionarse los componentes que intervienen en la generación de la respuesta en el servidor, es necesario que ya en el diseño de la aplicación se identifiquen claramente los objetos y sus interacciones; es decir, hay que modelar los objetos. Pero también es necesario identificar las páginas JSP o servlets que se necesitarán. Estás páginas se dividen habitualmente en dos grupos, uno correspondiente a las que manejan el flujo de la aplicación y la lógica que ello engloba, sin responsabilizarse de ningún tipo de presentación; es decir, estas páginas representan el punto de entrada a la aplicación web. El otro grupo de páginas es el encargado de generar el código HTML que se enviará al cliente; es decir, solamente contendrán indicaciones de presentación y lógica para presentar contenido dinámico.

La figura 1.8 refleja gráficamente la descripción que se ha expuesto en el párrafo anterior:

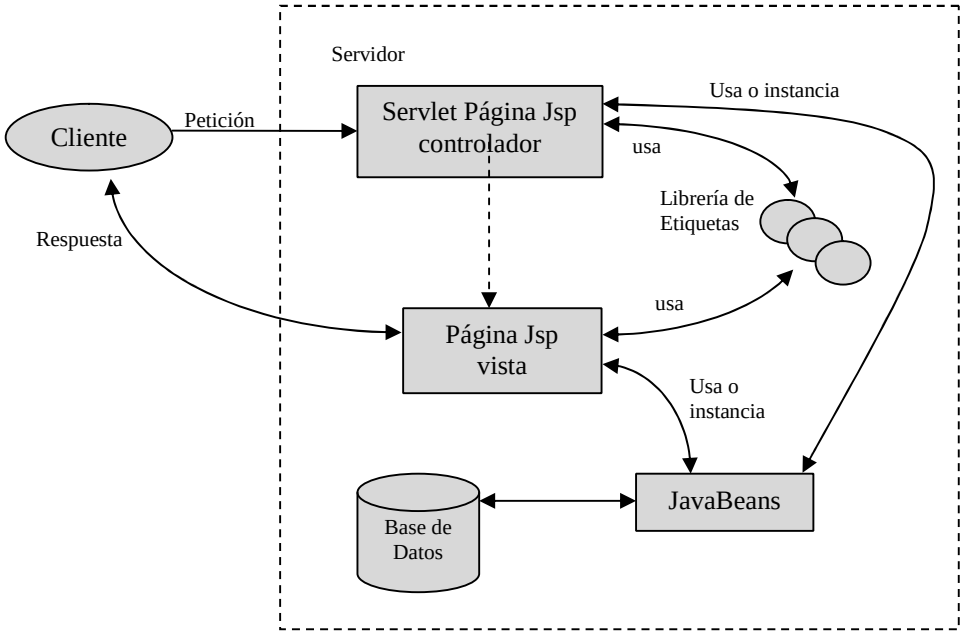


Figura 1.8: Modelo n-capas.

1.8. Bases de Datos Relacionales

1.8.1. Características

El modelo de bases de datos relacional es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. En seguida se definen los componentes de este modelo de bases de datos.

Una *Entidad* es cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, Persona, concepto abstracto o suceso. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. Hay dos tipos de entidades: *fuertes y débiles.* Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil [De Miguel 01].

Una *Relación (interrelación)* es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior. Las entidades que están involucradas en una determinada relación se denominan *entidades participantes*. El número de participantes en una relación es lo que determina el grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria, etc.

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

La cardinalidad con la que una entidad participa en una relación específica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial). Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio.

Los *atributos* son una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos. Gráficamente, se representan mediante círculos que cuelgan de las entidades o relaciones a las que pertenecen. Cada atributo tiene un conjunto de valores asociados denominado dominio. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio [De Miguel 01].

Un *identificador* de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad. Un identificador de una entidad debe cumplir dos condiciones [Silberschatz 02]:

- No pueden existir dos ocurrencias de la entidad con el mismo valor del identificador.
- Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.

Toda entidad tiene al menos un identificador y puede tener varios identificadores alternativos.

Además en este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

1.8.2.Normalización

La normalización es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional, desarrollada por E.F. Codd en 1972. Es una estrategia de diseño de abajo a arriba: se parte de los atributos y éstos se van agrupando en relaciones (tablas) según su afinidad. Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos
- No establece restricciones artificiales en la estructura de datos

Uno de los conceptos fundamentales en la normalización es el de dependencia funcional. Una dependencia funcional es una relación entre atributos de una misma relación (tabla). Si X e Y son atributos de la relación R, se dice que Y es funcionalmente dependiente de X (se denota por $X \rightarrow Y$) si cada valor de X tiene asociado un solo valor de Y (X e Y pueden constar de uno o varios atributos). A X se le domina determinante, ya que X determina el valor de Y. Se dice que el atributo Y es completamente dependiente de X si depende funcionalmente de X y no depende de ningún subconjunto de X.

La dependencia funcional es una noción semántica. Si hay o no dependencias funcionales entre atributos no lo determina una seria abstracta de reglas, si no, más bien, los modelos mentales del usuario y las reglas de negocio de la organización o empresa para la que se desarrolla el sistema de información. Cada dependencia funcional es una clase especial de regla de integridad.

En el proceso de normalización se debe ir comprobando que cada relación (tabla) cumple una seria de reglas que se basan en la clave primaria y las dependencias funcionales. Cada regla que se cumple aumenta el grado de normalización. Si una regla no se cumple, la relación se debe descomponer en varias relaciones que si la cumplan.

La normalización se lleva a cabo en una serie de pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización las relaciones tiene un formato más estricto (más fuerte) y, por lo tanto, son menos vulnerables a las anomalías de actualización. El modelo relacional solo requiere un conjunto de relaciones en primera forma normal. Las restantes formas normales son opcionales. Sin embargo, para evitar las anomalías de actualización, es recomendable llegar al menos a la tercera forma normal.

Primera forma normal (1FN). Una relación esta en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

Si una relación no esta en 1FN, hay que eliminar de ella los grupos repetitivos. Un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación. Hay dos formas de eliminar los grupos repetitivos. En la primera, se repiten los atributos con un solo valor para cada valor del grupo repetitivo. De este modo, se introducen redundancias ya que se duplican valores, pero estas redundancias se eliminaran después mediante las restantes formas normales. La segunda forma de eliminar los grupos repetitivos consiste en poner cada uno de ellos en una relación aparte, heredando la clave primaria de la relación en la que se encontraba.

Segunda forma normal (2FN). Una relación esta en segunda forma normal si y solo si está en 1FN y, además, cada atributo no primo (que no esta en la clave primaria) es completamente dependiente de la clave primaria.

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación esta en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también esta en 2FN. Las relaciones que no están en 2FN pueden sufrir anomalías cuando se realizan actualizaciones.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de la que dependen).

Tercera Forma Normal (3FN). Una relación esta en tercera forma normal si, y solo si, está en 2FN y, además, cada atributo no primo no depende transitivamente de la clave primaria. La dependencia $X \rightarrow Z$ es transitiva si existen las dependencias $X \rightarrow Y$, $Y \rightarrow Z$, siendo X, Y, atributos o conjuntos de atributos de una misma relación.

Aunque las relaciones en 2FN tienen menos redundancias que las relaciones en 1FN, todavía pueden sufrir anomalías frente a las actualizaciones. Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

Forma Normal de Boyce - Codd (BCFN). Una relación esta en la forma normal de Boyce – Codd si, y solo si, todo determinante es una clave candidata.

La 2 FN y la 3FN eliminan las dependencias parciales y las dependencias transitivas de la clave primaria. Pero este tipo de dependencias todavía pueden existir sobre otras claves candidatas, si estas existen. La BCFN es más fuerte que la 3FN, por lo tanto, toda relación en BCFN está en 3FN.

La violación de la BCFN es poco frecuente ya que se da bajo ciertas condiciones que raramente se presentan. Se debe comprobar si una relación viola la BCFN si tiene dos o más claves candidatas compuestas que tienen al menos un atributo en común.

El análisis del sistema es la etapa en la que se realiza una investigación acerca del problema que se pretende resolver y cómo puede ser resuelto. En esta fase del desarrollo, la comunicación entre los analistas, los programadores y los usuarios es de gran importancia para que el sistema cumpla con las expectativas planteadas.

El primer paso en la etapa de análisis es definir el problema, una vez hecho, se indica qué necesita ser resulto; en este capítulo realizaremos dicho proceso.

2.7. Definición del problema.

En este proyecto se requiere realizar una aplicación Web que ayude a controlar las reservaciones de un salón de fiestas, y que cumpla con las siguientes características:

- Tener la alternativa de crear paquetes personalizados.
- Realizar la cotización mientras se crea el paquete
- Verificar la disponibilidad de fecha en el calendario.
- Notificar vía e-mail la reservación realizada.
- Controlar las altas, bajas y modificaciones de usuarios, productos, tipos de evento, clasificaciones de productos así como las reservaciones generadas.

2.8. Objetivos generales y específicos del proyecto.

Objetivos Generales

- Desarrollar un sistema de información basado en web que proporcione cotizaciones de paquetes de eventos para un salón de fiestas.
- El sistema deberá cumplir con las características de seguridad, robustez, calidad, rapidez y escalabilidad.

Objetivos Específicos

- Elaborar la especificación de requerimientos del proyecto
- Realizar el análisis de los requerimientos usando UML con Visual Paradigm.
- Diseñar la base de datos.
- Diseñar la aplicación basada en web usando UML.
- Implementar los diseños realizados usando el lenguaje orientado a objetos Java.
- Realizar pruebas del sistema.

2.9. Identificación de Usuarios.

Con la definición del problema podemos identificar los diferentes tipos de usuarios que interactúan con el sistema:

- Cliente que dispone de un visualizador.
- Administrador de la Base de Datos (Puede ser el mismo que el administrador del Sitio Web).
- Administrador del Sitio Web (Puede ser el mismo que el administrador de la Base de Datos).

El *cliente*. va a poder crear un paquete totalmente personalizado para su evento (boda, quince años, bautizo, entre otros) dependiendo de sus necesidades, durante el proceso de personalización del paquete podrá consultar la cotización del mismo; podrá ir agregando, cambiando o bien eliminando los diferentes platillos dentro de su paquete. También podrá consultar las sugerencias que contienen los menús más solicitados donde se define la entrada, el platillo fuerte, y las opciones de postre.

También puede realizar la solicitud de reservación enviando sus datos personales, características del paquete personalizado, fecha, duración del evento y número de personas, al igual podrá solicitar la cancelación de reservación si éste lo desea realizando un contacto directo con el administrador.

El *Administrador*. Las tareas que puede desempeñar son las mismas que un cliente y aparte otras tareas para el manejo del sistema, por ejemplo dará de alta el tipo de evento, clasificación de productos, alta de productos y usuarios, también podrá consultar y modificar las clasificaciones, tipos de evento, productos y reservaciones. En lo que se refiere a modificación de reservaciones únicamente se podrá modificar el estado de la reservación (activa o cancelada) y el número de personas. Si el cliente desea realizar cambio de fecha o de los platillos, el administrador realizará primero la cancelación de la reservación actual y posteriormente dará de alta una nueva con las características que se deseen.

El administrador es el único que puede realizar una cancelación validando los datos de la reservación que se le proporcionan al cliente cuando la realiza.

Como el administrador puede consultar las reservaciones tiene acceso a una agenda que constantemente estará actualizando con la finalidad de llevar un control de los eventos por efectuar.

A continuación se describirá el proceso de identificación de las funciones del sistema a desarrollar a través de los casos de uso.

2.10. Modelado de Casos de Uso.

El modelado de Casos de Uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione. Es una manera muy buena de dirigirse hacia el análisis de sistemas orientado a objetos. Los casos de uso son generalmente el punto de partida del análisis orientado a objetos con UML.

Cada caso de uso se documenta por una descripción del escenario. La descripción puede ser escrita en modo de texto o en un formato paso a paso. Cada caso de uso puede ser también definido por otras propiedades, como las condiciones pre- y post- del escenario condiciones que existen antes de que el escenario comience, y condiciones que existen después de que el escenario se completa.

En seguida se muestra el esquema general de los casos de uso para el sistema de reservaciones (Figura 2.1) y posteriormente se describe el modelado de casos de uso más relevantes:

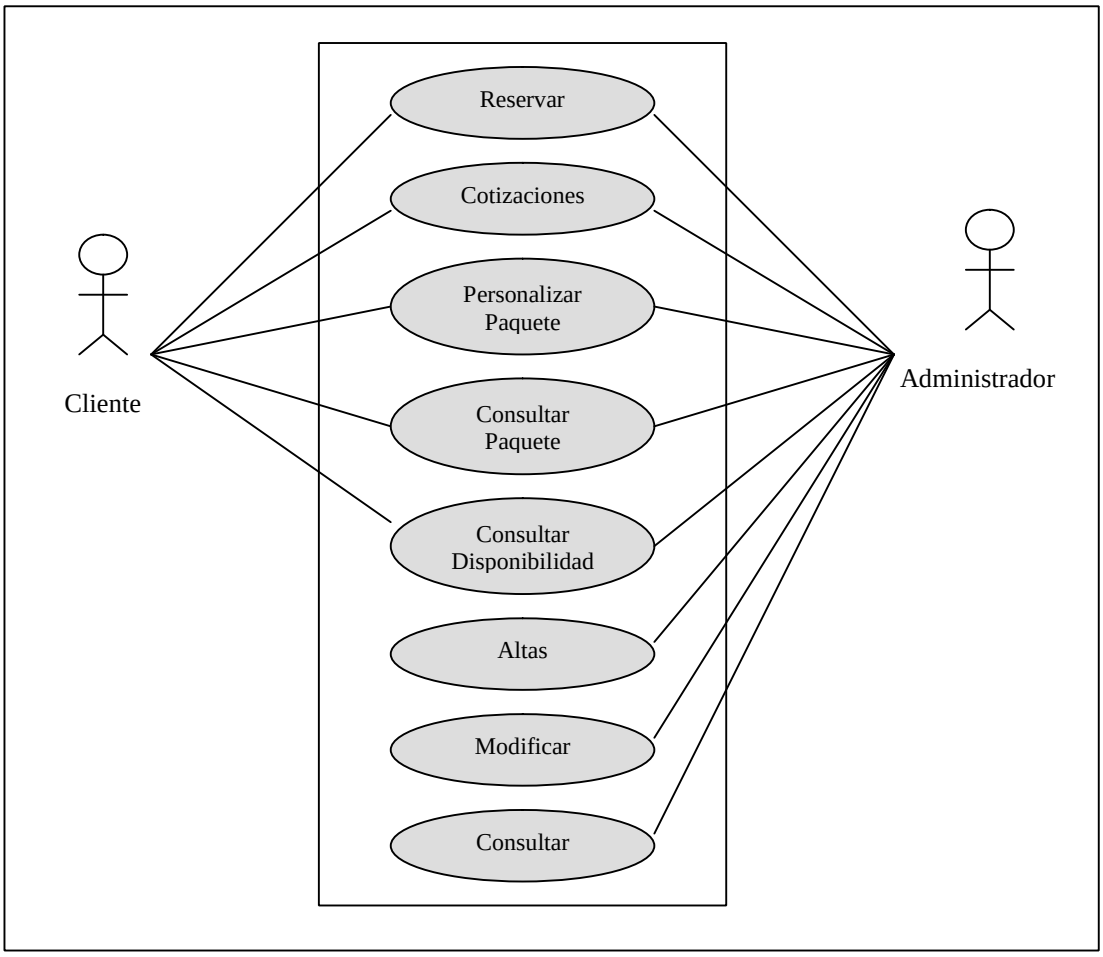


Figura 2.1 Esquema General de Casos de Uso.

Caso de uso: Reservación de salón.

- **Actores:** usuario.
- **Propósito:** realizar la reservación de un salón de fiestas para un evento.
- **Resumen:** cliente ingresa a la página principal, selecciona la opción de reservar, el sistema pide información para verificar si se podrá llevar a cabo la reservación en la fecha, hora y duración que el cliente requiere, si la fecha está disponible se elijen los platillos de 3 tiempos y el número de personas, después visualiza el costo y el cliente puede realizar aún cambios en su elección o bien continuar, después tendrá que ingresar sus datos personales y elegir enviar, por último se envía notificación a su email con los datos de la reservación que en ese momento solicito.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa página principal y elije reservación	**2.** solicita fecha, hora y duración del evento.
3. elije fecha, hora y duración de su evento	**4.** verifica datos de disponibilidad del evento
	5. si hay disponibilidad se pedirá que elija su menú de 3 tiempos y número de personas
6. elije platillos de 3 tiempos y número de personas	**7.** muestra el costo total
8. elije reservar	**9.** pide datos personales del cliente
10. ingresa información personal	**11.** realiza la reservación
	12. envía email al cliente con las características de su reservación
13. visualiza notificación de email	

- **Escenario de extensiones alternativas:**

Acción de los actores	Respuestas del Sistema
1. ingresa página principal y elije opción reservación	**2.** solicita fecha, hora y duración del evento.
3. elije fecha, hora y duración de su evento	**4.** verifica datos de disponibilidad, no hay
	5. muestra al cliente que esa fecha u hora ya se encuentran reservadas
6. visualiza notificación del sistema	

Caso de uso: Consultar Contacto.

- **Actores:** usuario.
- **Propósito:** consultar contacto con el salón.
- **Resumen:** cliente ingresa a la página principal, elije la opción contáctanos, se muestra información acerca de la dirección del lugar, teléfono y dirección de correo electrónico.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción contáctanos	**4.** muestra información del salón.
5. da click en liga de correo	
6. solicita información que desee	
7. envía mail y notifica su envío	

- **Escenario de extensiones alternativas:** no aplica.

Caso de uso: Cancelación de reservación.

- **Actores:** Administrador.
- **Propósito:** realizar la cancelación de un evento.
- **Resumen:** ingresa a la página principal selecciona opción administrador e ingresa su usuario y contraseña, en caso de ser correctos selecciona la opción para modificar reservaciones, el sistema le muestra todas las reservaciones, se elije la deseada y en ese momento se edita, y se concluye cambiando el estado de la reservación por cancelado.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos correctos
7. elije modificar reservación	**8.** muestra todas las reservaciones
9. selecciona el número de la reservación que se desea modificar.	**10.** muestra datos de la reservación
11. cambia estado a cancelado	**12.** guarda cambios y envía notificación
13. visualiza notificación	

- **Escenario de extensiones alternativas:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos incorrectos
	7. notifica que sus datos no son correctos
8. visualiza notificación	

Caso de uso: Agregar Producto.

- **Actores:** administrador.
- **Propósito:** agregar un nuevo producto.
- **Resumen:** ingresa a la página principal, elige opción administrador, pide identificarse, si es correcta la información selecciona alta producto, se pide ingresar la descripción del producto, precio y la clasificación por último oprime el botón agregar.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos correctos y muestra menú de administrador
7. elije alta producto	**8.** muestra campos de captura
9. ingresa información del nuevo producto como descripción, clasificación y precio.	**10.** guarda información y notifica la acción.
11. visualiza notificación.	

- **Escenario de extensiones alternativas:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos incorrectos
	7. notifica que sus datos no son correctos
8. visualiza notificación	

Caso de uso: Consulta Reservaciones.

- **Actores:** administrador.
- **Propósito:** consultar reservaciones.
- **Resumen:** ingresa a la página principal, pide la opción administrador, el sistema solicita identificación, una vez que se ingresa los datos de usuario si se validan exitosamente elije consultar reservación aparece un formato que muestra todas las reservaciones y si deseamos ver más información de alguna de ellas se da clic en el número de reservación para que muestre toda la información al respecto.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos correctos y muestra menú de administrador
7. elije consultar reservaciones	**8.** muestra todas las reservaciones
9. da clic en el número de la reservación que desee consultar	**10.** muestra toda la información de la reservación que seleccionó
11. visualiza datos	

- **Escenario de extensiones alternativas:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos incorrectos
	7. notifica que sus datos no son correctos
8. visualiza notificación	

Caso de uso: Modificar Producto.

- **Actores:** administrador.
- **Propósito:** Modificar productos.
- **Resumen:** ingresa a la página principal, elije administrador e ingresa datos correctos, selecciona modificar producto, aparece una lista de todos los productos, en esta parte podrá cambiar no solo el precio de cualquier producto sino también su descripción y el tipo de clasificación de uno o varios productos a la vez y finalmente guarda los cambios.
- **Tipo:** primario
- **Escenario principal de éxito:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos correctos, muestra menú de administrador.
7. elije modificar producto	**8.** muestra la lista de productos.
9. modifica cuantos productos desee	**10.** guarda los cambios y notifica la operación
11. visualiza notificación.	

- **Escenario de extensiones alternativas:**

Acción de los actores	Respuestas del Sistema
1. ingresa a la página principal	**2.** muestra menú general.
3.selecciona opción administrador	**4.** solicita usuario y contraseña
5. ingresa datos	**6.** valida datos incorrectos
	7. notifica que sus datos no son correctos
8. visualiza notificación	

2.11. Diseño Conceptual de la BD.

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo de esta etapa es comprender:

- ✓ La perspectiva que cada usuario tiene de los datos.
- ✓ La naturaleza de los datos, independientemente de su representación física.
- ✓ El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad- relación. En la figura 2.2 se construye utilizando la información que se encuentra en la especificación de los requerimientos del usuario.

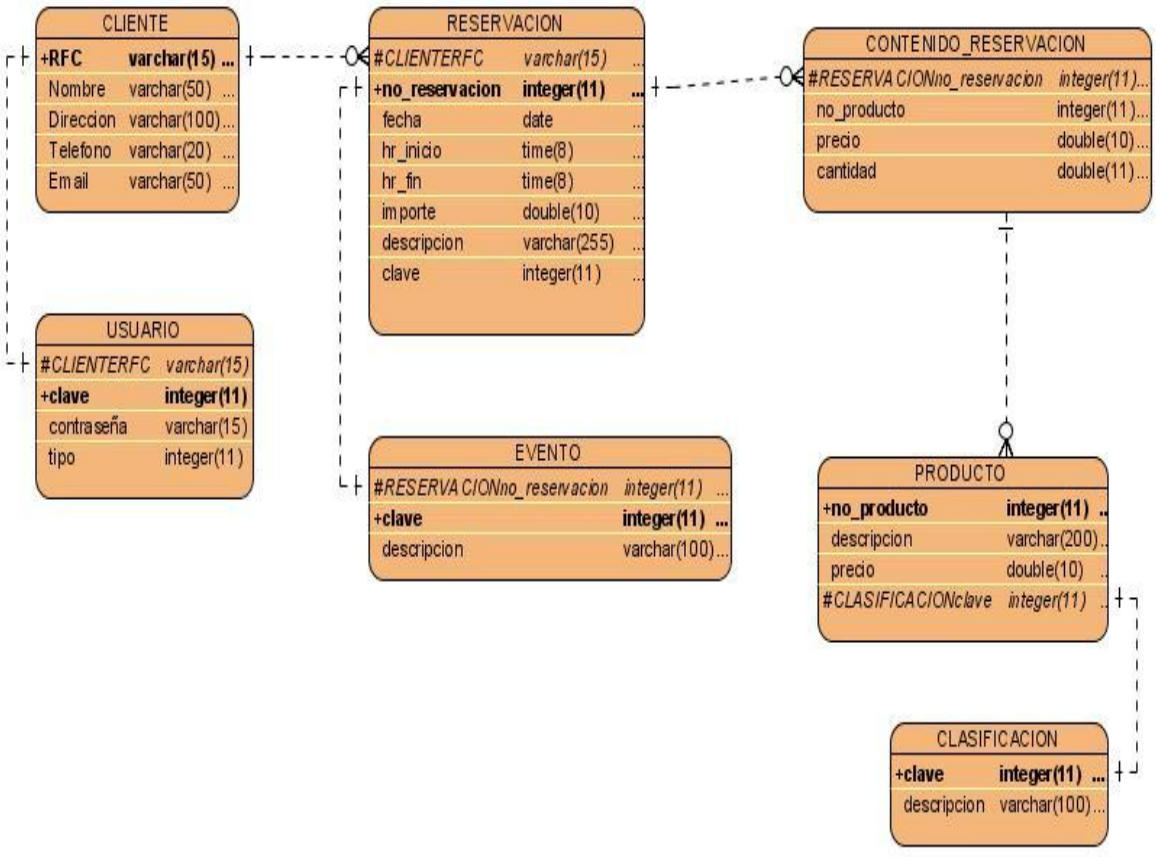


Figura 2.2 Diagrama Entidad - Relación

2.12. Diagrama de Flujo de la Información.

De acuerdo a la especificación de requerimientos, se identificaron las funciones primordiales del sistema y posteriormente con su análisis funcional se desglosaron cada una de las funciones, en seguida se mostraran los diagramas de flujo de la información más importantes.

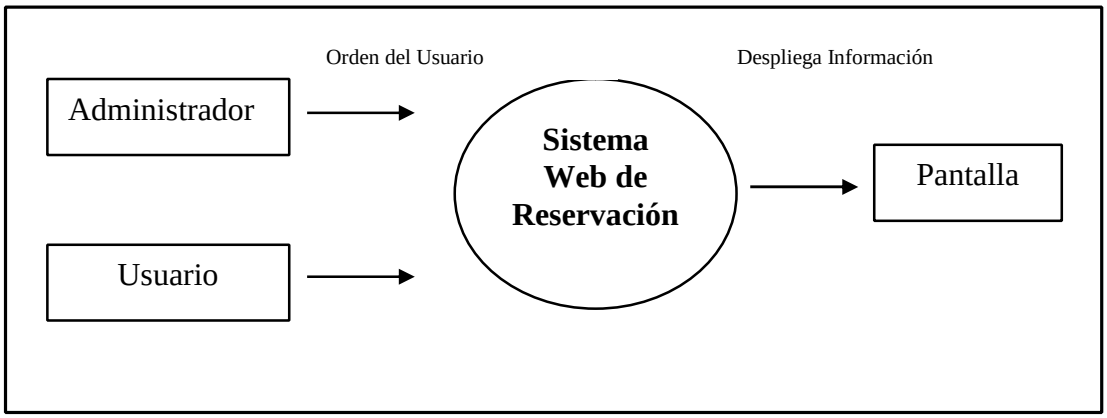


Figura 2.3. DFD nivel 0

La Figura 2.3 muestra como el usuario o administrador envían una orden al sistema Web, el cuál procesa esta orden y muestra la respuesta desplegando la información solicitada en pantalla.

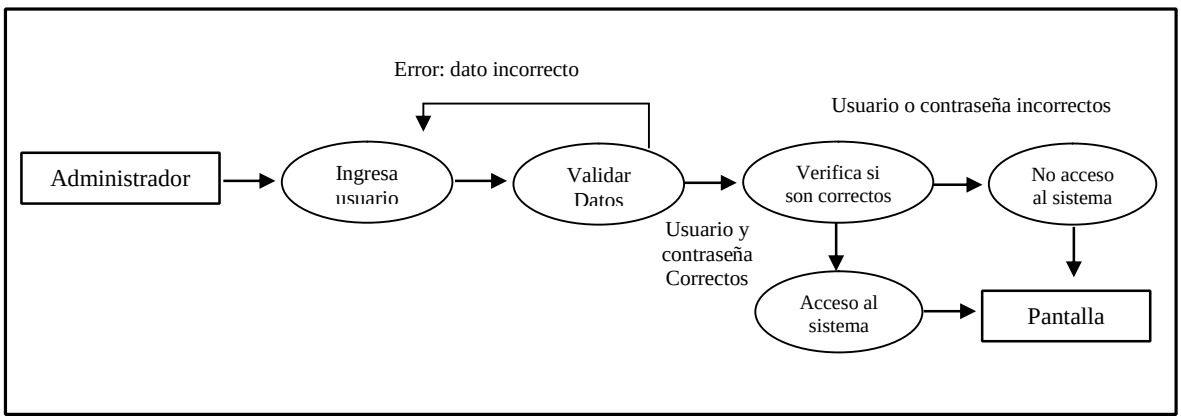


Figura 2.4. DFD nivel 1.1

La Figura 2.4 muestra cómo el administrador accede al sistema Web. El administrador debe proporcionar su usuario y contraseña, los cuales serán validados para poder ingresar al sistema, en caso de que algún dato sea incorrecto se notificará a través de la pantalla al igual que en caso de que sus datos sean aceptados se mostrara en pantalla el menú del administrador.

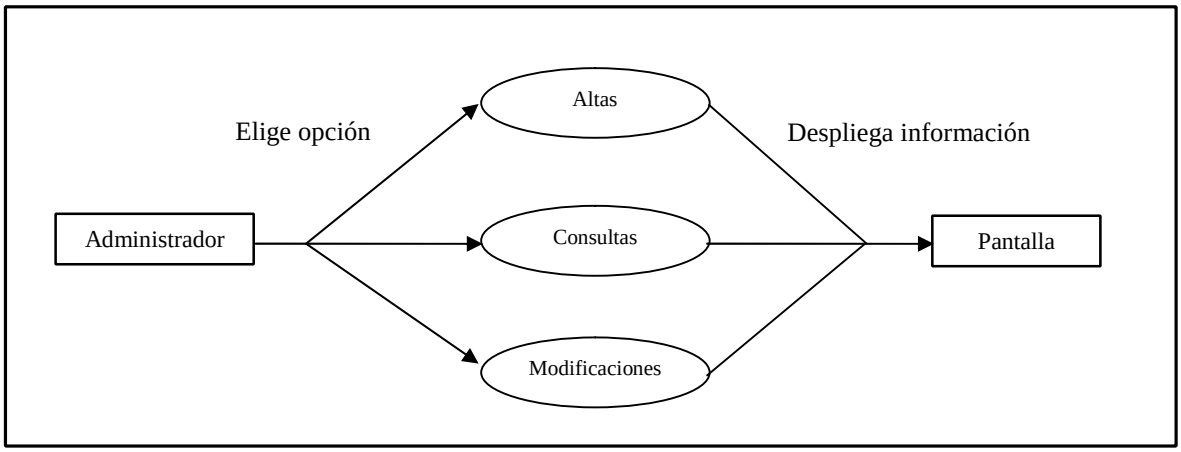


Figura 2.5. DFD nivel 1.1.1

La Figura 2.5 muestra los tres módulos del menú del administrador (altas, consultas y modificaciones), y solamente el administrador puede tener acceso a ellos.

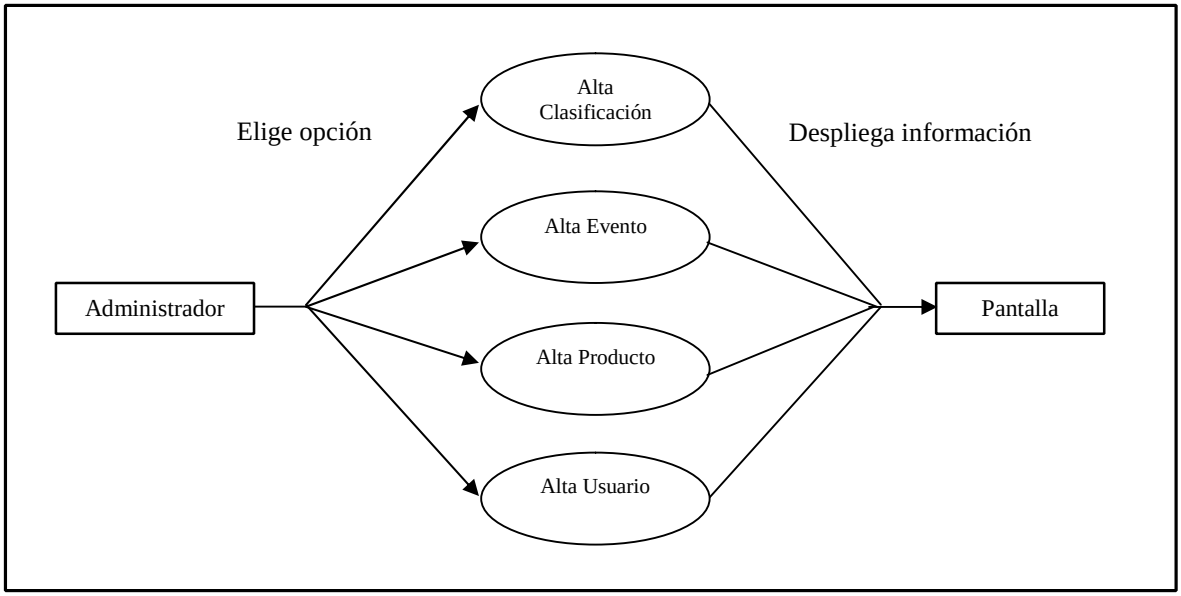


Figura 2.6. DFD nivel 1.1.1.1

La Figura 2.6 se observan las funciones del módulo Altas, dicho módulo consta de 4 apartados fundamentales que son: alta de clasificación, alta de evento, alta de producto y alta de usuario.

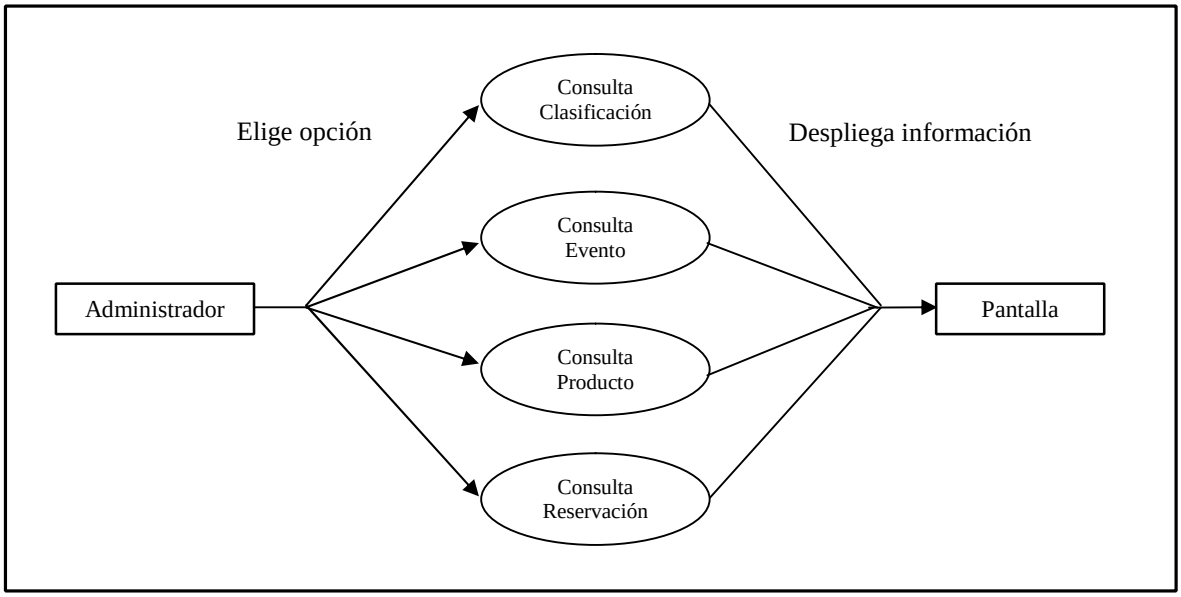


Figura 2.7. DFD nivel 1.1.1.2

La Figura 2.7 se observan las funciones del módulo Consultas, dicho módulo consta de 4 apartados fundamentales que son: consulta de clasificación, consulta de evento, consulta de producto y consulta de reservación.

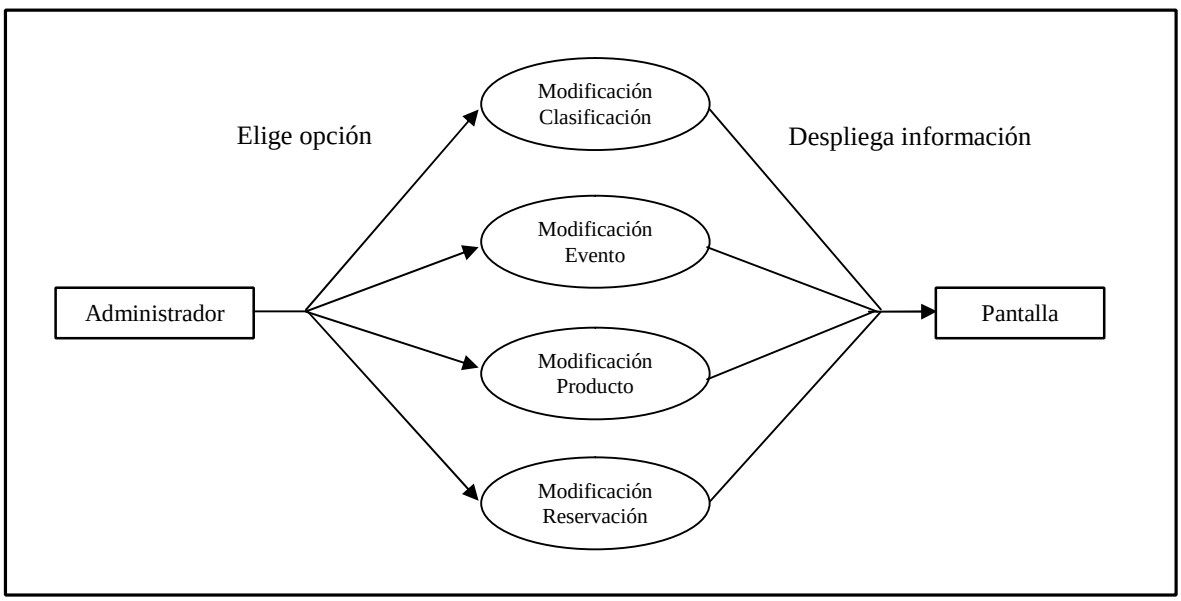


Figura 2.8. DFD nivel 1.1.1.3

La Figura 2.8 se observan las funciones del módulo Modificaciones, dicho módulo consta de 4 apartados fundamentales que son: modificación de clasificación, modificación de evento, modificación de producto y modificación de reservación.

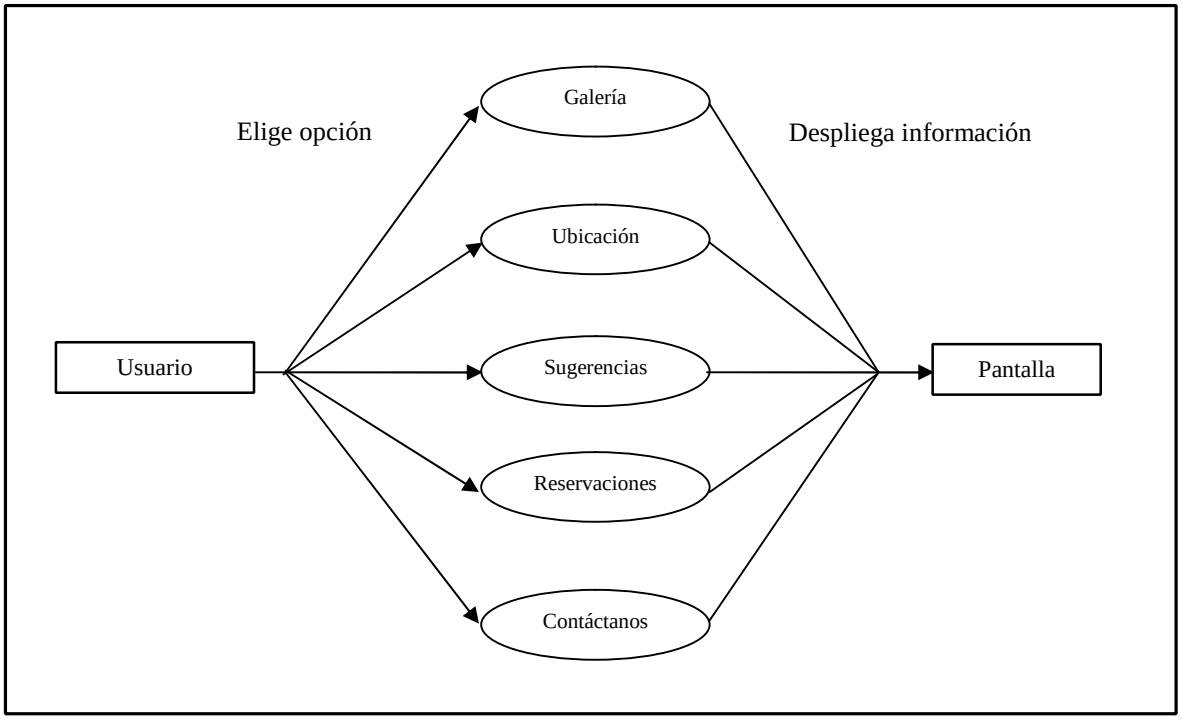


Figura 2.9. DFD nivel 1.2

En la Figura 2.9 se observan los módulos a los que puede acceder el usuario, cabe mencionar que también el administrador puede tener acceso aparte de sus módulos especiales.

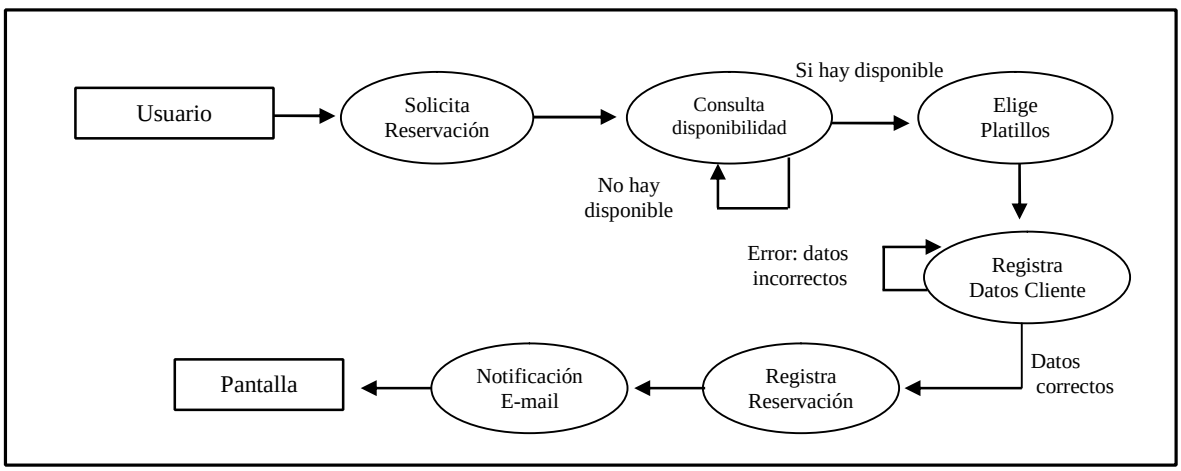


Figura 2.10. DFD nivel 1.2.4

En la Figura 2.10 se describe el flujo de los datos que sigue el usuario para llevar a cabo una reservación.

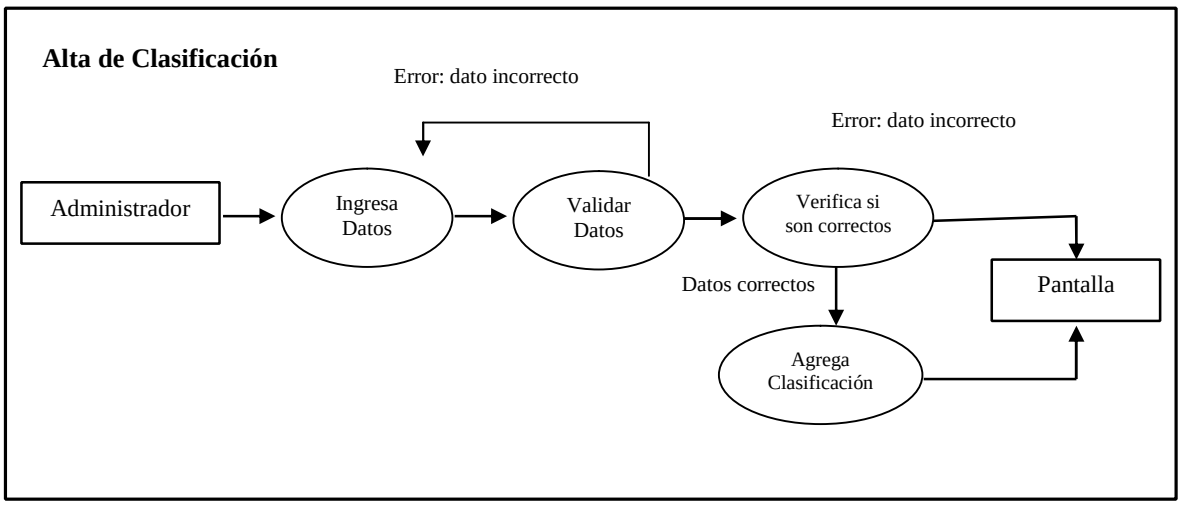


Figura 2.11. DFD nivel 1.1.1.1.1

En la Figura 2.11 se muestra como el administrador da de alta una clasificación, ingresando los datos (descripción), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

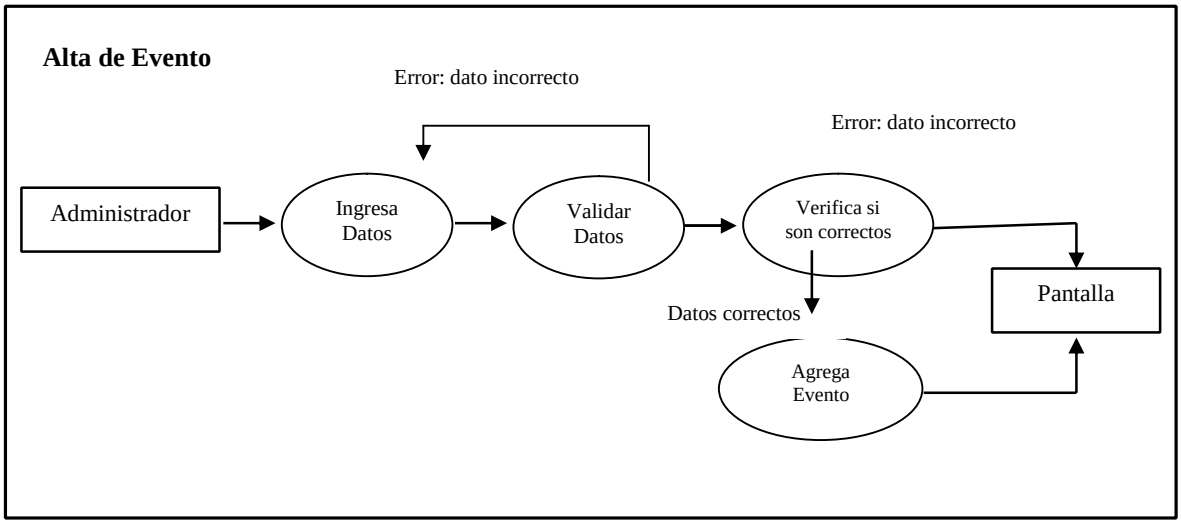


Figura 2.12. DFD nivel 1.1.1.1.2

En la Figura 2.12 se muestra como el administrador da de alta un evento, ingresando los datos (descripción), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

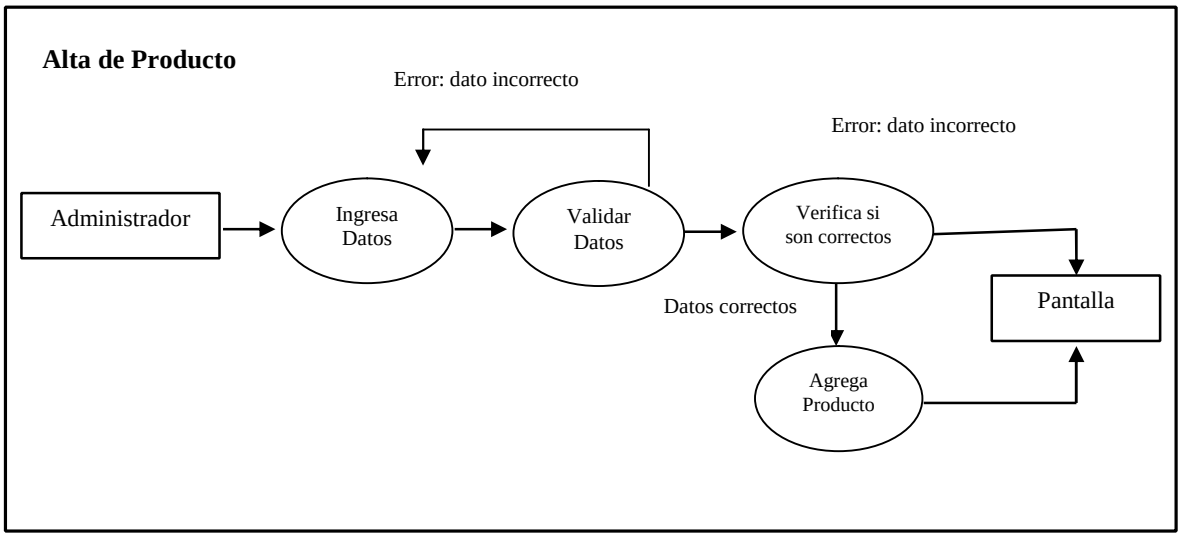


Figura 2.13. DFD nivel 1.1.1.1.3

En la Figura 2.13 se muestra como el administrador da de alta un producto, ingresando los datos (descripción, precio y clasificación), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

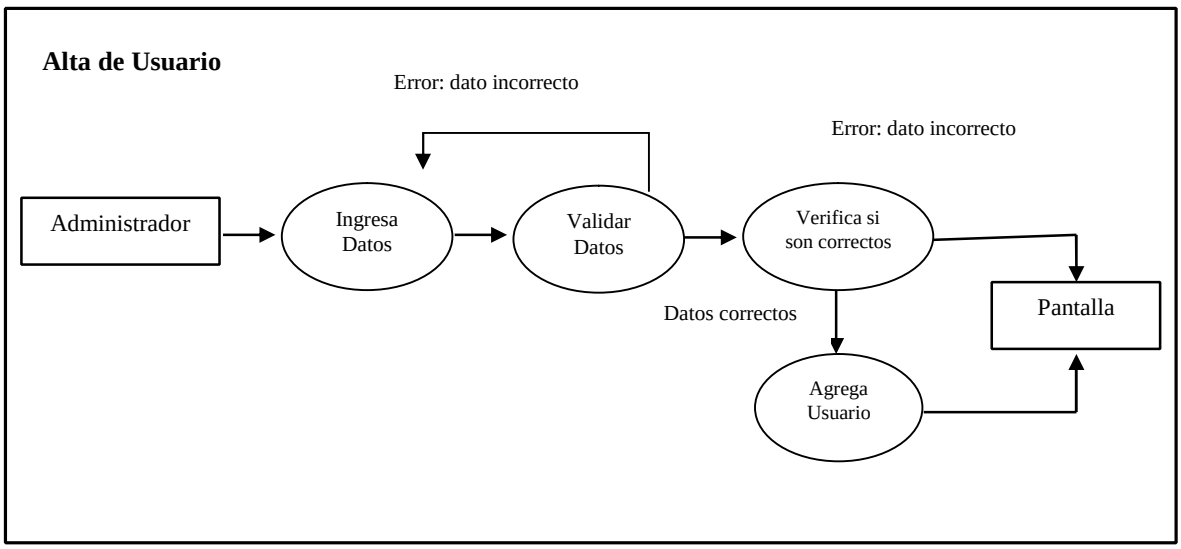


Figura 2.14. DFD nivel 1.1.1.1.4

En la Figura 2.14 se muestra como el administrador da de alta un usuario, ingresando los datos (contraseña y tipo de usuario), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

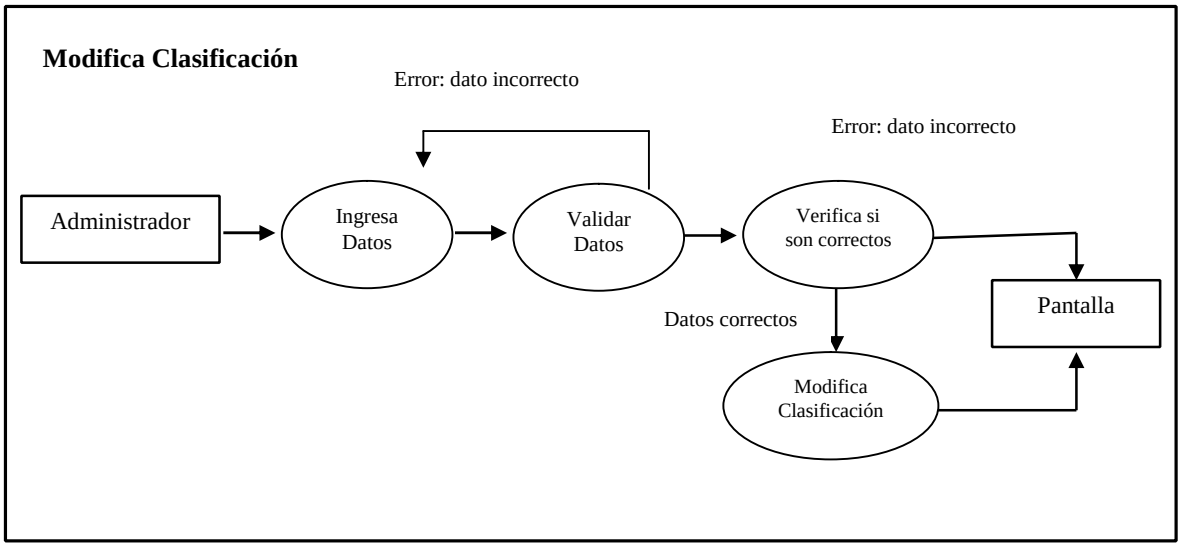


Figura 2.15. DFD nivel 1.1.1.3.1

En la Figura 2.15 se muestra como el administrador modifica una clasificación, ingresando los datos (descripción), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

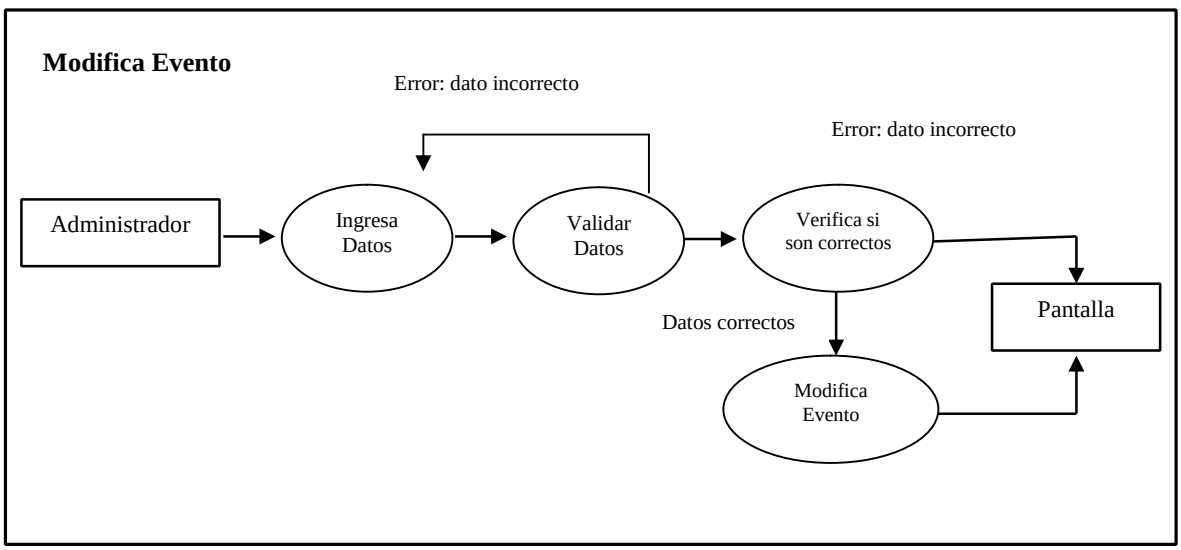


Figura 2.16. DFD nivel 1.1.1.3.2

En la Figura 2.16 se muestra como el administrador modifica un evento, ingresando los datos (descripción), la aplicación valida la información, si es correcta lo agrega y se visualiza la notificación.

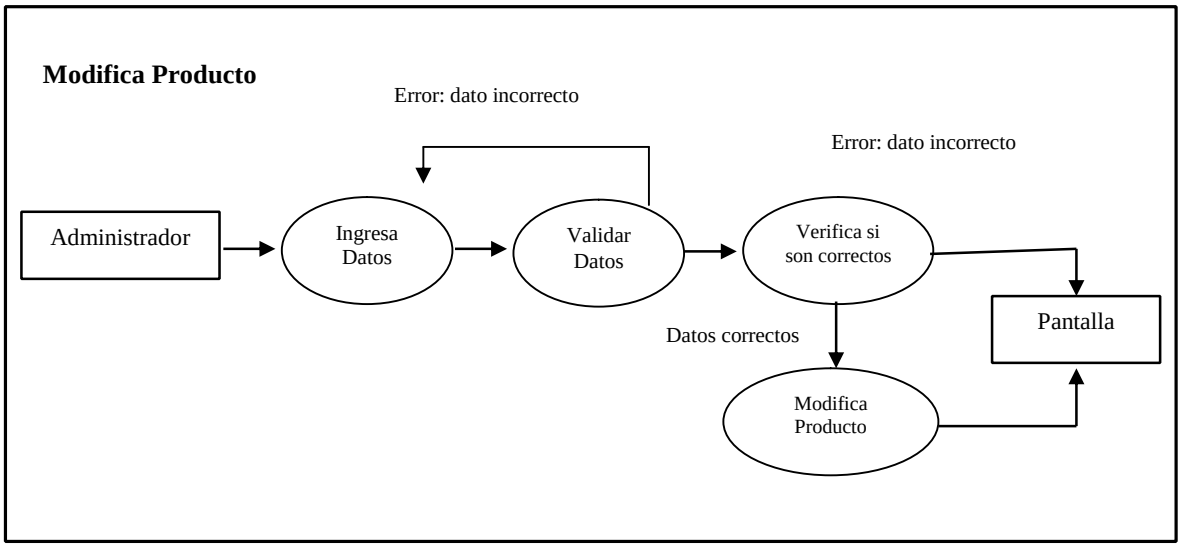


Figura 2.17. DFD nivel 1.1.1.3.3

En la Figura 2.17 se muestra como el administrador modifica un producto, ingresando los datos (descripción, precio y clasificación), la aplicación valida la información, si son correctos modifica el producto y se visualiza la notificación.

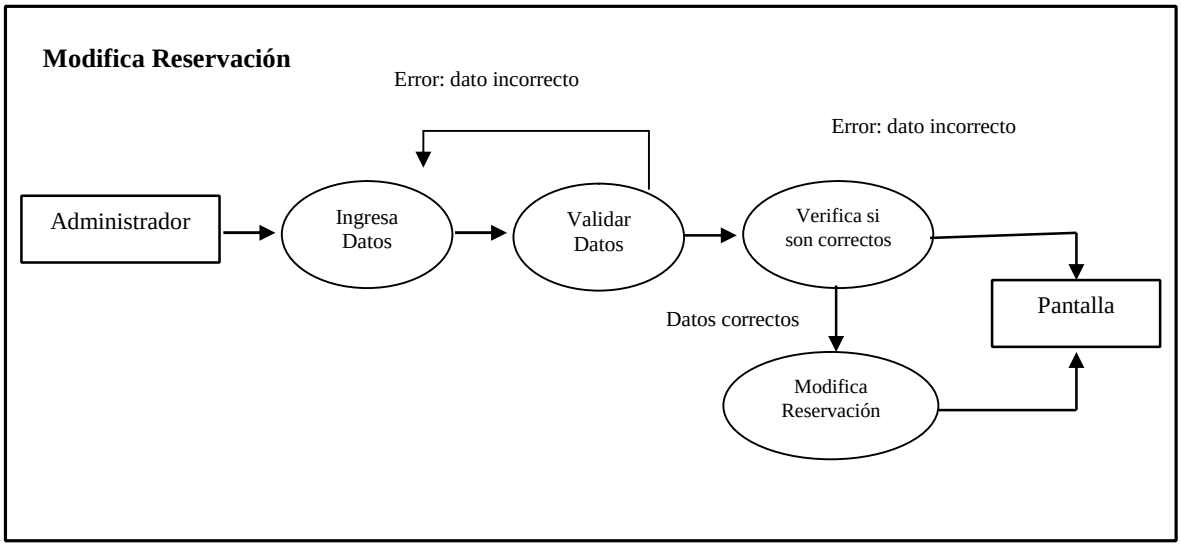


Figura 2.18. DFD nivel 1.1.1.3.4

En la Figura 2.18 se muestra como el administrador modifica una reservación, cambiando los datos (estado o número de personas), la aplicación valida la información, si son correctos el cambio es realizado.

En está fase ya comenzamos a diseñar el sistema aunque iremos progresivamente incorporando información al diseño. Seguiremos una serie de pasos hasta la obtención del diseño completo del sistema, durante este capítulo nos apoyaremos en diagramas de estado, comunicación, clases, interacción y el diseño lógico de base de datos.

3.5. Identificación de Entidades.

A continuación se listan y describen las entidades identificadas.

Usuario.
Son los usuarios que tienen los privilegios para llevar a cabo tanto la administración del sitio web, como de la base de datos. A demás tienen acceso al sitio web como un cliente.

Cliente.
Son los usuarios que tienen acceso a la página del salón elite donde pueden realizar reservaciones en línea para sus eventos sociales, además de consultar información acerca de la ubicación y contactos del mismo.

Producto.
Son los productos que ofrece el salón elite, en esta aplicación solo se agregaron alimentos como productos posteriormente se puede ampliar el uso de esta entidad.

Clasificación.
Es la entidad que se encarga de clasificar los diferentes productos que se manejan, principalmente en los platillos describe a que categoría corresponde cada uno de ellos.

Evento.
En la entidad evento se almacena los diferentes tipos de eventos que ofrece nuestro sitio web.

Reservación.
En esta entidad se va a almacenar de forma general la información acerca de las reservaciones del salón elite.

Contenido_reservación.
Es la entidad que almacena el paquete personalizado de cada una de las reservaciones.

3.6. Diccionario de datos.

El presente diccionario muestra la forma en que la información alfanumérica se ha estructurado y descrito conceptualmente para poder ser ingresada a la base de datos. Se describe cada una de las entidades y sus atributos, lo cual es necesario para una mejor interpretación de la información.

Entidades:

- ✓ Usuario
- ✓ Cliente
- ✓ Producto
- ✓ Clasificación
- ✓ Evento
- ✓ Reservación
- ✓ Contenido_reservación

CLIENTE (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
RFC	Identificador del cliente	Primary Key	varchar	15	a-z, A-Z, 0-9	Combinación de estos caracteres	No
Nombre	Nombre del cliente	Multivalorado	varchar	50	a-z, A-Z	Combinación de estos caracteres	No
Dirección	Domicilio del cliente	Multivalorado	varchar	100	a-z, A-Z, 0-9	Combinación de estos caracteres	No
Teléfono	Teléfono del cliente	Multivalorado	varchar	20	0-9, -, ()	Combinación de estos caracteres	Si
Email	Email del cliente	Multivalorado	varchar	100	a-z, A-Z, 0-9, _,@,.	Combinación de estos caracteres	Si

Tabla 3.1 Entidad Cliente

USUARIO (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
Clave	Identificador del usuario	Primary Key	Int	11	0-9	Auto_increment	No
Contraseña	Contraseña del usuario	Multivalorado	Varchar	15	a-z, A-Z, 0-9	Combinación de estos caracteres	No
Tipo	Tipo de usuario	Multivalorado	Int	11	0-1	Cero o uno	No

Tabla 3.2 Entidad Usuario

PRODUCTO (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
No_producto	Identificador del producto	Primary Key	int	11	0-9	Combinación de estos caracteres	No
Clv_clasificacion	Identificador de la clasificación	Foreign Key	int	11	0-9	Combinación de estos caracteres	No
Descripción	Descripción del producto	Monovalorado	varchar	200	a-z, A-Z, 0-9, .,	Combinación de estos caracteres	No
Precio	Costo del producto	Multivalorado	double	10	0-9, .	Combinación de estos caracteres	

Tabla 3.3 Entidad Producto

CLASIFICACION (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
Clave	Identificador de la clasificación	Primary Key	Int	11	0-9	Combinación de estos caracteres	No
Descripción	Descripción de la clasificación	Monovalorado	varchar	100	a-z, A-Z, 0-9, .,	Combinación de estos caracteres	No

Tabla 3.4 Entidad Clasificación

EVENTO (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
Clave	Identificador del evento	Primary Key	Int	11	0-9	Combinación de estos caracteres	No
Descripción	Descripción del evento	Monovalorado	Varchar	100	a-z, A-Z, 0-9, .,	Combinación de estos caracteres	No

Tabla 3.5 Entidad Evento

RESERVACION (Entidad Fuerte)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
No_reservacion	Número de la reservación	Primary Key	int	11	0-9	Combinación de estos caracteres	No
RFC	Identificador del cliente	Foreign Key	varchar	15	a-z, A-Z, 0-9	Combinación de estos caracteres	No
Fecha	Fecha de la reservación	Multivalorado	date		Año-mes-día	Combinación de estos caracteres	No
Hr_inicio	Hora inicial del evento	Multivalorado	datetime		Año-mes-día hr:mm:ss	Combinación de estos caracteres	No
Hr_fin	Hora final del evento	Multivalorado	datetime		Año-mes-día hr:mm:ss	Combinación de estos caracteres	No
Importe	Costo de la reservación	Multivalorado	double	10	0-9, .	Combinación de estos caracteres	No
Clave	Clave del evento	Foreign Key	int	11	0-9	Combinación de estos caracteres	No
Estado	Estado de la reservación	Multivalorado	Varchar	20	Activa o Cancelado	Alguno de estos valores	No
Descripción	Descripción de la reservación	Multivalorado	Varchar	200	a-z, A-Z, 0-9, .,	Combinación de estos caracteres	No

Tabla 3.6 Entidad Reservación

CONTENIDO_RESERVACION (Entidad Débil)							
ATRIBUTOS	**DESCRIPCION**	**TIPO DE ATRIBUTO**	**TIPO DE DATO**	**LONGITUD**	**DOMINIO**	**DESCRIPCION DE DOMINIO**	**NULO**
No_reservacion	Número de la reservación	Foreign Key	Int	11	0-9	Combinación de estos caracteres	No
No_producto	Identificador del producto	Foreign Key	Int	11	0-9	Combinación de estos caracteres	No
Precio	Precio del producto	Multivalorado	Doublé	10	0-9, .	Combinación de estos caracteres	No
Cantidad	Número de personas	Multivalorado	Int	11	0-9	Combinación de estos caracteres	No

Tabla 3.7 Entidad Contenido Reservación

3.7. Diagrama de Estados.

Los diagramas de estado describen gráficamente los eventos y los estados de los objetos. Los diagramas de estado son útiles, entre otras cosas, para indicar los eventos del sistema en los casos de uso.

Un *evento* es un acontecimiento importante a tomar en cuenta para el sistema. Un *estado* es la condición de un objeto en un momento determinado: el tiempo que transcurre entre eventos. Una *transición* es una relación entre dos estados, e indica que, cuando ocurre un evento, el objeto pasa del estado anterior al siguiente.

En UML, los estados se representan mediante un rectángulo con los bordes redondeados. Las transiciones se representan mediante flechas con el nombre del evento respectivo. Se acostumbra poner un estado inicial (círculo negro) y un estado final (circulo negro dentro de otro círculo).

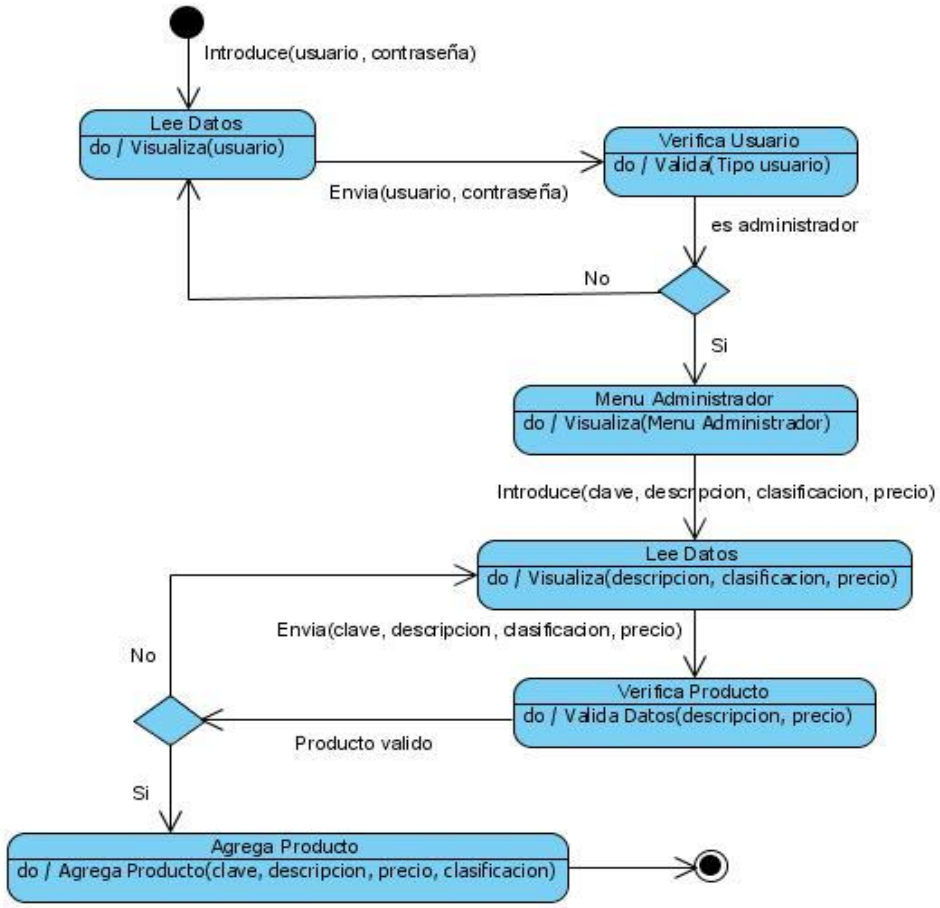


Figura 3.1 Diagrama de estado para agregar productos.

En la figura 3.1 se observan los diferentes estados del sistema para que el administrador pueda dar de alta un producto. Primero se valida si el usuario que realizará la operación tiene los permisos

para hacerlo, posteriormente se visualiza el menú del administrador, se ingresa la clave, descripción, clasificación y precio del producto a agregar, después pasa al estado de verificar producto, donde se verifica que los datos ingresados sean correctos para finalmente agregar el producto a la base de datos.

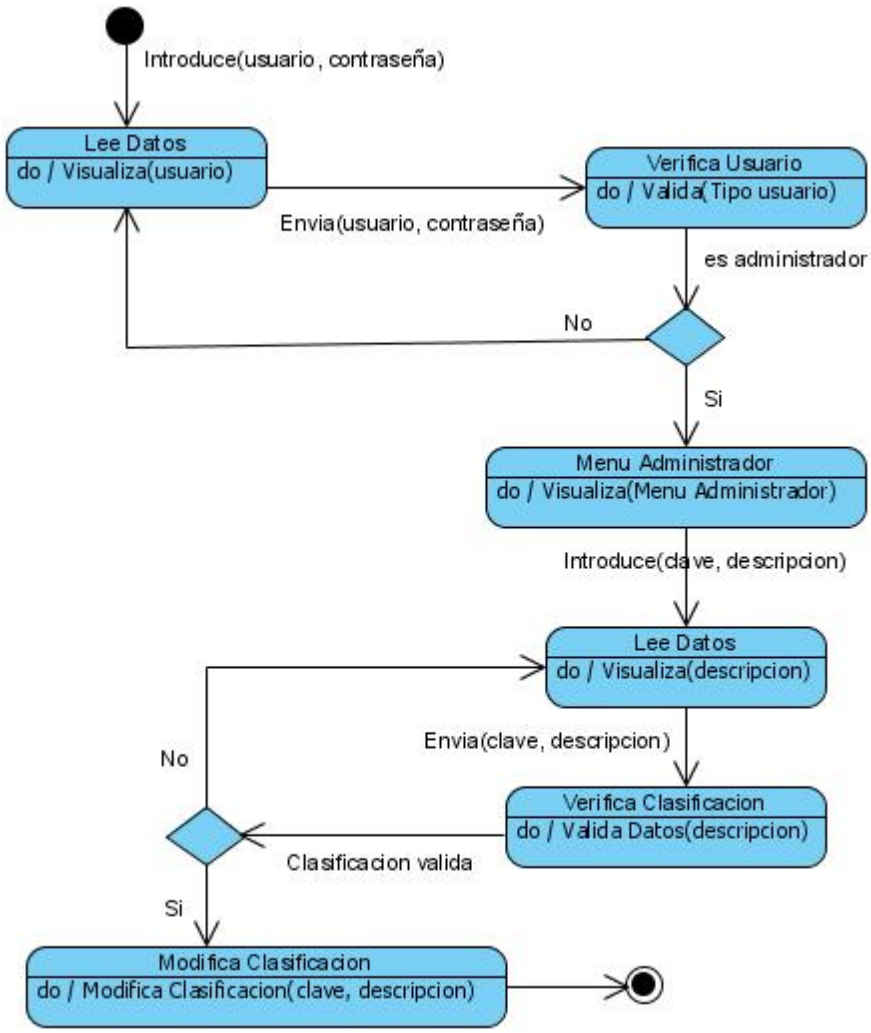


Figura 3.2 Diagrama de estado para modificar clasificaciones.

En la figura 3.2 se observan los diferentes estados del sistema para que el administrador pueda modificar la clasificación de un producto. Primero se valida si el usuario que realizará la operación tiene los permisos para hacerlo, posteriormente se visualiza el menú del administrador, se modifican los campos deseados y después pasa al estado de verificación en donde se comprueba que sean correctos; finalmente, se lleva a cabo la modificación en la base de datos.

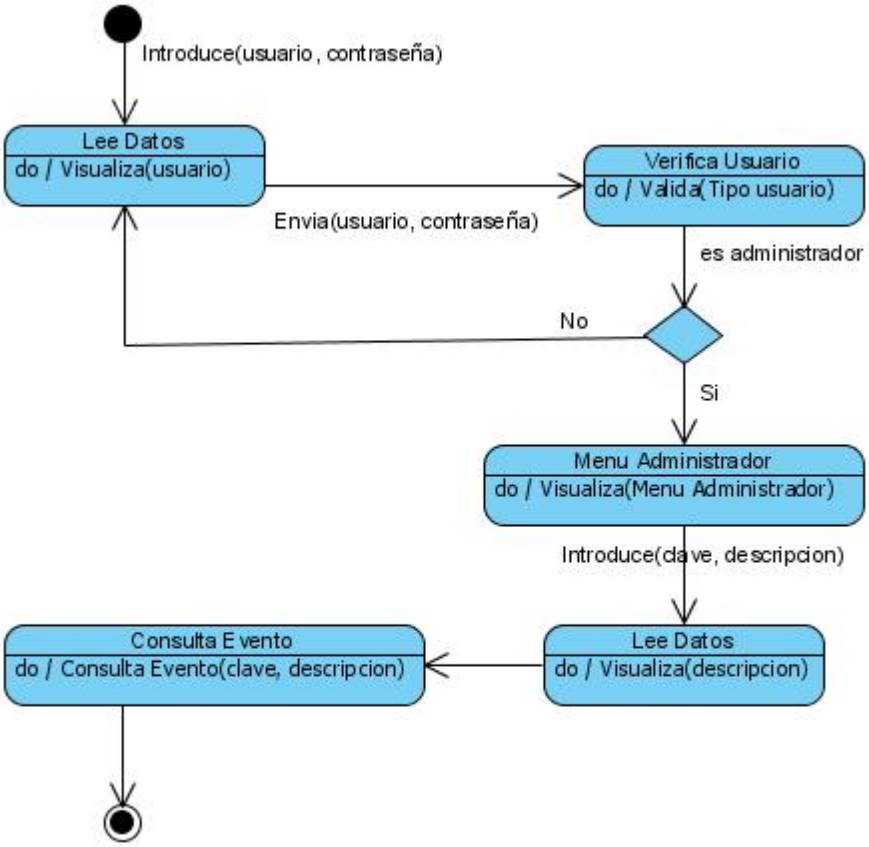


Figura 3.3 Diagrama de estados para consultar eventos.

En la figura 3.3 se muestran los estados que recorre el sistema para que el administrador consulte los tipos de eventos. Inicialmente se valida que el tipo de usuario que ingresa al sistema pueda realizar esta tarea, en seguida se visualiza el menú del administrador; finalmente, se muestra un listado con la información solicitada.

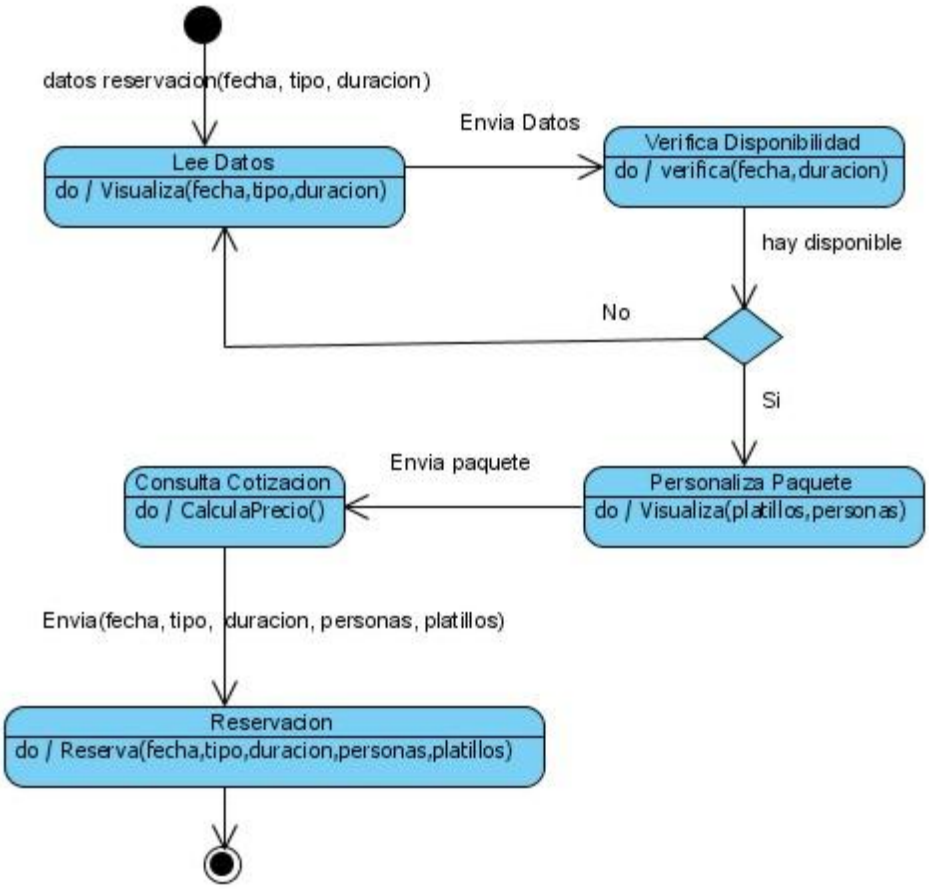
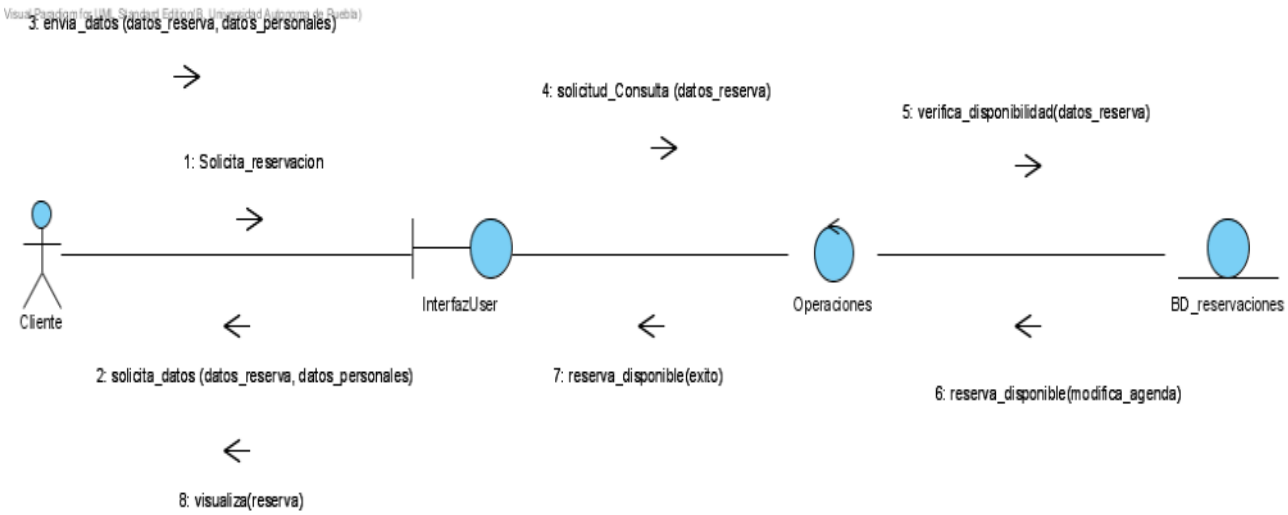


Figura 3.4 Diagrama de estados para reservaciones.

En la figura 3.4 podemos observar el diagrama de estados para que el usuario registre una reservación. Inicialmente elije la fecha, tipo y duración de su evento, después se verifica la disponibilidad de las fechas seleccionadas, si están disponibles se continúa a la personalización del paquete, posteriormente se visualiza el costo total; por último se procede con el registro de la reservación.

3.7.1. Diagrama de Comunicación.

Un diagrama de comunicación muestra la colaboración dinámica entre los elementos. Es similar al diagrama de secuencia y la intención es para enfocar como los objetos colaboran entre ellos. También muestran los intercambios de mensajes o (interacciones) entre los objetos.



El diagrama de comunicación ocupa las flechas para indicar el flujo de los mensajes entre objetos, las etiquetas son puestas sobre las flechas para mostrar el orden de los mensajes. En el siguiente diagrama podremos observar la comunicación entre las interfaces existentes para que se lleve a cabo una reservación óptima.

Figura 3.5 Diagrama de comunicación para reservaciones.

En la figura 3.6 se presenta el proceso de comunicación para la reservación NO óptima.

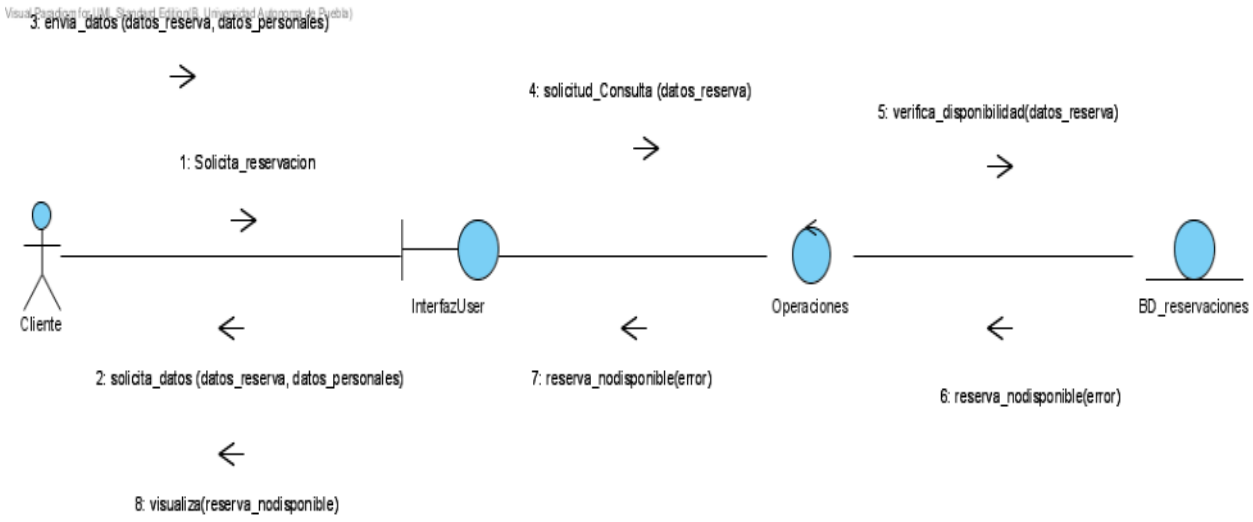


Figura 3.6 Diagrama de comunicación para reservaciones NO óptima.

3.7.2. Diagrama de Clases.

Los diagramas de clase se encargan de clasificar los objetos y sus relaciones. A través del proceso de casos de usos nos apoyamos a identificar las clases. Las clases de objetos se modelan utilizando diagramas de estructura o clases que muestran la estructura general del sistema, así como las propiedades relacionales y de comportamiento. En un diagrama de clases, los objetos que forman parte del sistema se agrupan en clases y cada una de éstas tiene contiene una sección de nombre y otra de atributos. Algunas clases también incluyen una sección de operaciones, que específica como se deben comportar los objetos de dicha clase.

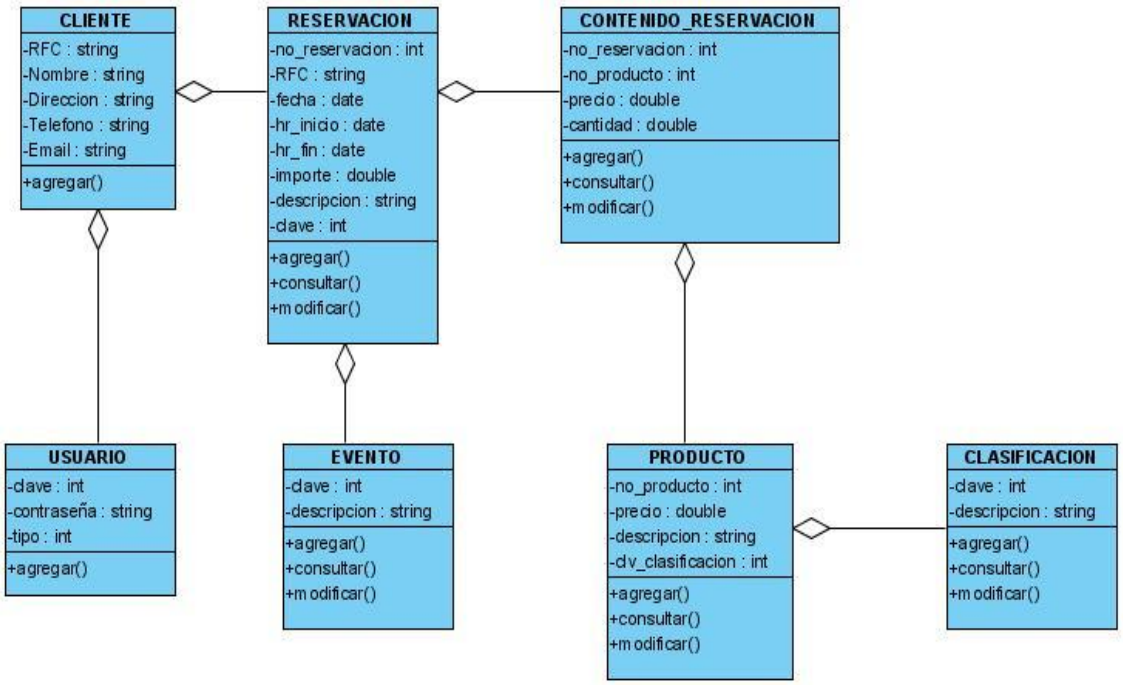


Figura 3.7 Diagrama de Clases.

3.7.3. Diagrama de Interacción.

El diagrama de interacción, representa la forma en cómo un Cliente (Actor) u Objetos (Clases) se comunican entre si en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. Los componentes de dicho diagrama son: Un Objeto Actor, Mensaje de un objeto a otro objeto, Mensaje e un objeto así mismo.

En seguida se agregan diferentes diagramas de interacción de algunas de las actividades más importantes de este sistema.

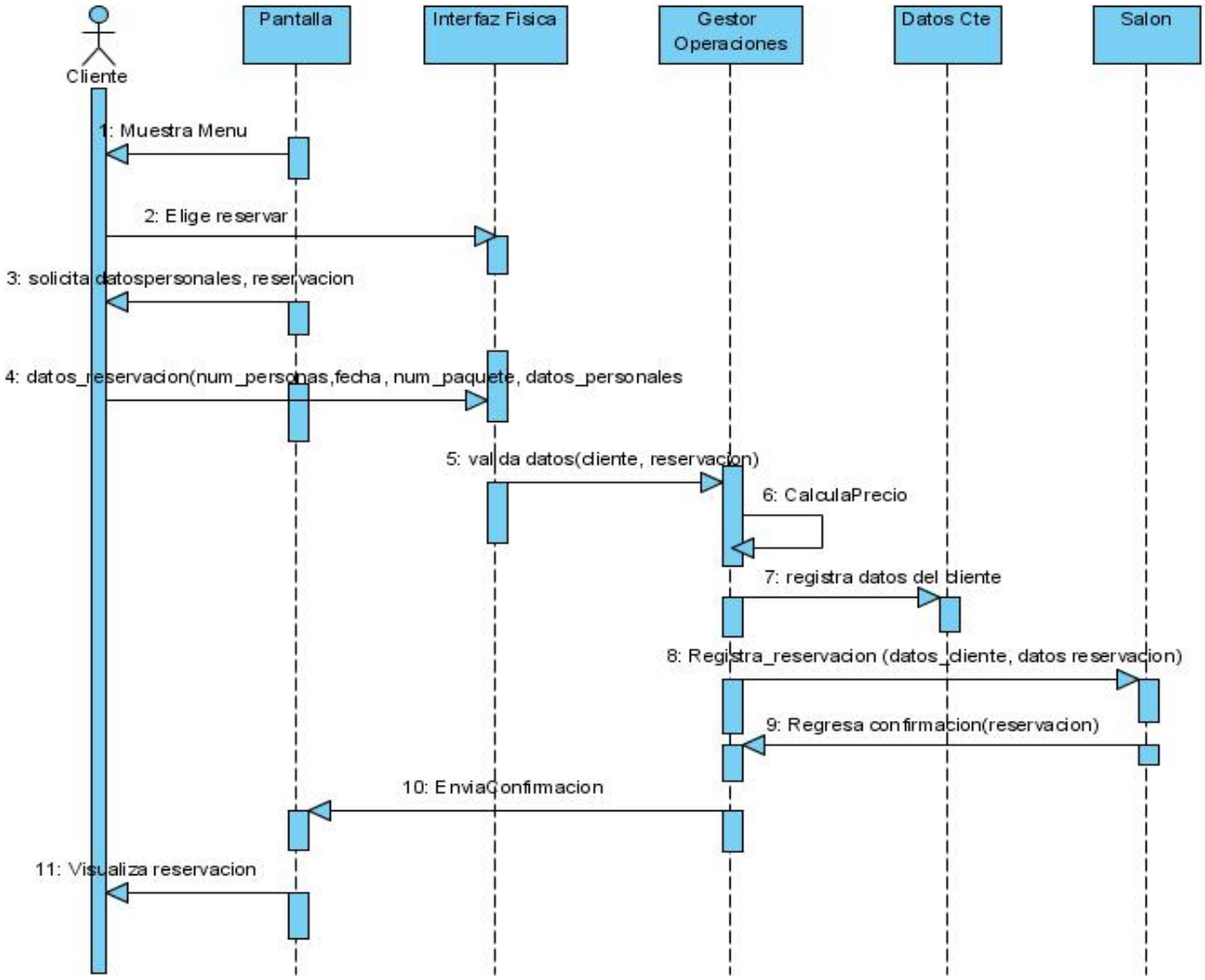


Figura 3.8 Diagrama de interacción para solicitar reservación.

En la figura 3.8 se puede identificar como interactúa el usuario con las diferentes interfaces del sistema cuando desea realizar la reservación para un evento. Primero la página mostrara un menú, el

cliente selecciona la opción de reservación ocupando la interfaz física, inmediatamente se solicita información acerca de la reservación: fecha, hora, tipo de evento y duración; después el gestor de operaciones valida la disponibilidad de fecha; si hay disponibilidad se procede a personalizar el paquete y se calcula el costo total; el gestor de operaciones solicita los datos del cliente y registra la reservación; por último se envía una notificación al cliente indicando que la operación se realizo con éxito.

3.8. Diseño Lógico.

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, en esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizarán las estructuras de datos del modelo de bases de datos como el modelo relacional. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario. El diseño lógico y el conceptual son las etapas clave para conseguir que un sistema funcione correctamente. Si el esquema no es una representación fiel de la empresa, no será difícil, sino imposible, definir todas las vistas de usuario, o mantener la integridad de la base de datos.

En la tabla 3.8 se muestra el diseño lógico que se ha obtenido del sistema:

Entidad	Atributos
Usuario	**PK** clave, contraseña, tipo
Cliente	**PK** RFC, nombre, dirección, teléfono, email
Cliente_usuario	**FK** RFC, **FK** clave
Reservación	**PK** no_reservacion, **FK** RFC, fecha, hr_ inicio, hr_fin, importe, descripción, clave, estado
Producto	**PK** no_producto, **FK** clv_clasificacion, precio, descripción
Evento	**PK** clave, descripción
Contenido_reservacion	**FK** no_reservacion, **FK** no_producto, precio, cantidad
Clasificacion	**PK** clave, descripción

Tabla 3.8 Diseño lógico del sistema

Una vez efectuado el análisis y diseño del sistema de reservaciones, se realizó la implementación; haciendo uso de un sistema administrador de base de datos, un lenguaje de programación y algunas herramientas de diseño para la implementación del sistema.

4.4. Herramientas para el desarrollo del sistema.

4.4.1. Manejador de bases de datos.

El administrador de la base de datos utilizado es MySql. Es un manejador de base de datos relacional, multihilo y multiusuario. Usa el lenguaje estandarizado de SQL para el almacenamiento, actualización y acceso a los datos. MySQL es una aplicación de Código abierto y por lo tanto es gratuita, nos permite redistribuir una aplicación que la contenga y nos permite incluso modificar su código para mejorarla o adaptarla a nuestras necesidades.

4.4.2. Lenguaje de programación.

El lenguaje de programación utilizado es Java (versión 1.4.1) creado por *Sun Microsystems.*

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

- ✓ Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- ✓ Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- ✓ Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

4.4.3. Herramientas de Diseño.

Se utilizó Dreamweaver y Flash 8 para el diseño de las interfaces web, Tomcat 5.0 como servidor de aplicaciones, JCreator como editor para los jsps.

4.4.4. Equipo utilizado.

Celeron 1.8Ghz
DD 80Gb
Memoria RAM 256 Mb
Sistema operativo Windows XP Profesional
Monitor 17’’
Mouse
Teclado

4.4.5. Equipo requerido.

Celeron 1.8Ghz
DD 40Gb
Memoria RAM 256 Mb
Sistema operativo Windows 2000 o superior
Monitor 15’’
Mouse
Teclado
Tarjeta de red o modem
Internet explorer 6.0 o superior

4.4.6. Definición de la base de datos.

A continuación se muestran imágenes de la implementación de la base de datos salón_fiestas con la herramienta MySQL Administrator.

Esquema general de la base de datos:

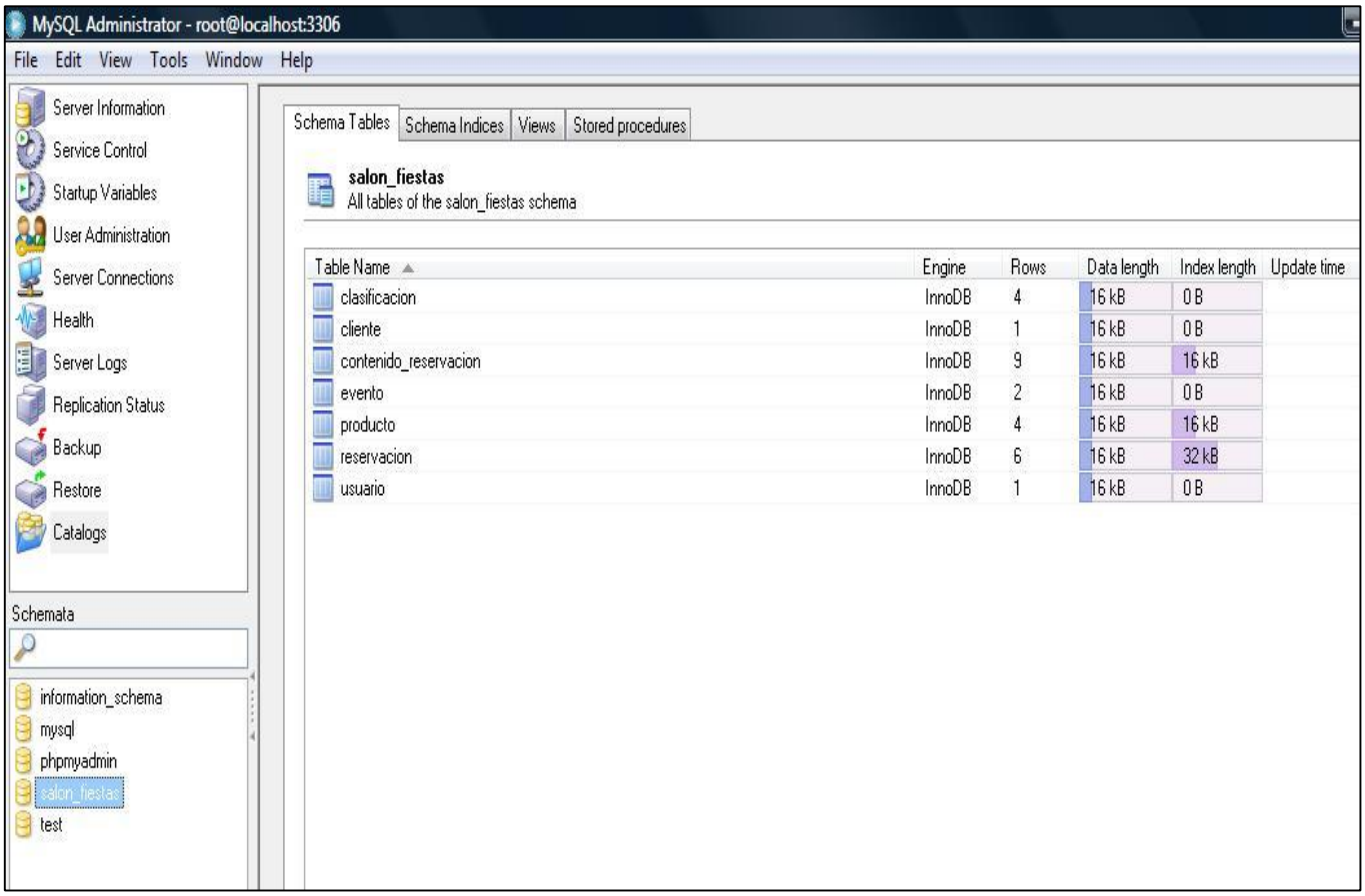


Figura 4.1 Esquema General

Definición de la tabla cliente:

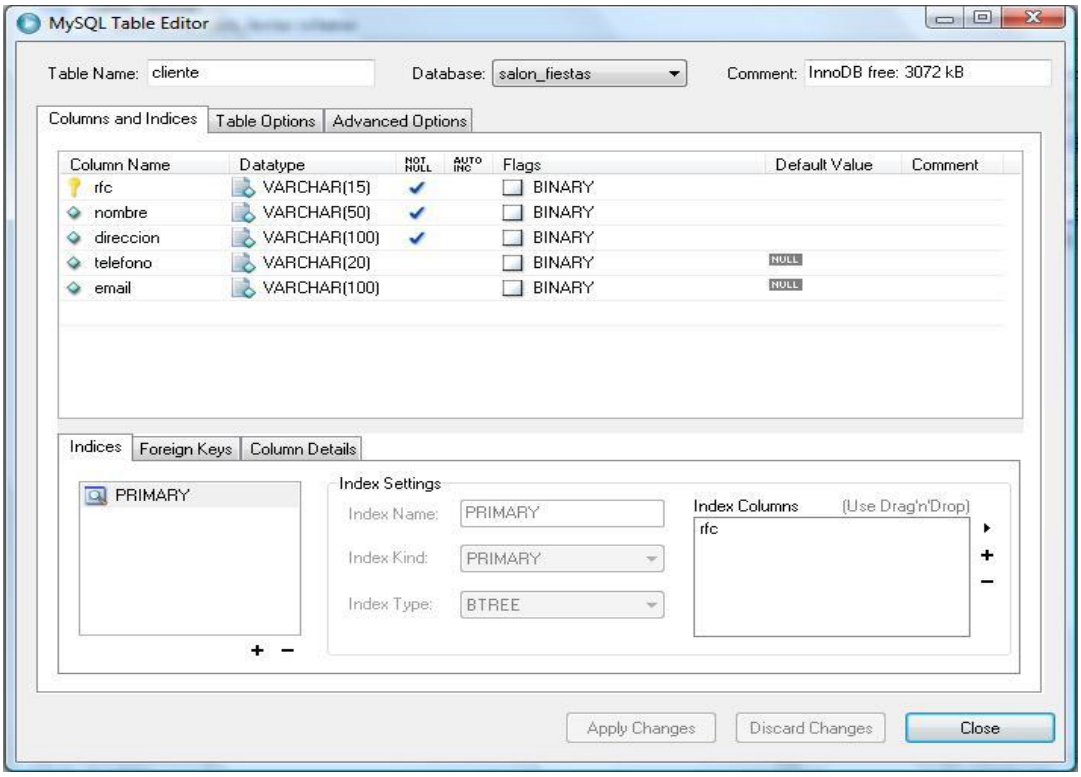


Figura 4.2 Tabla cliente

Definición de la tabla evento:

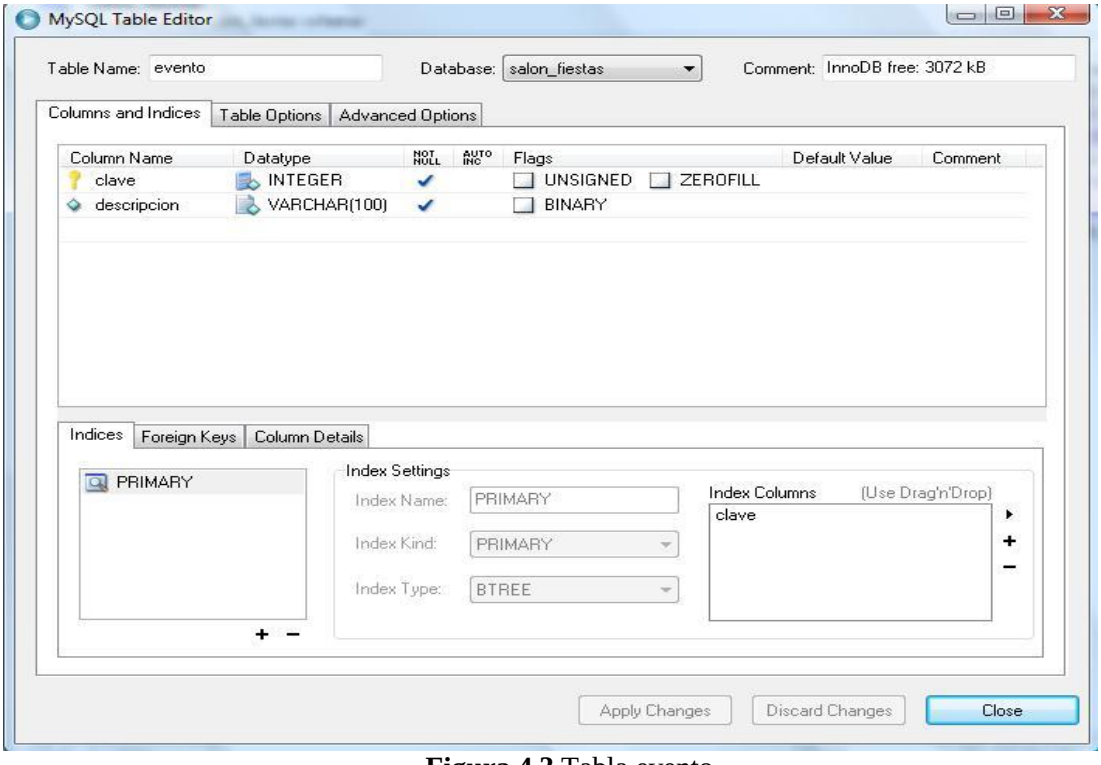


Figura 4.3 Tabla evento

Definición de la tabla reservación:

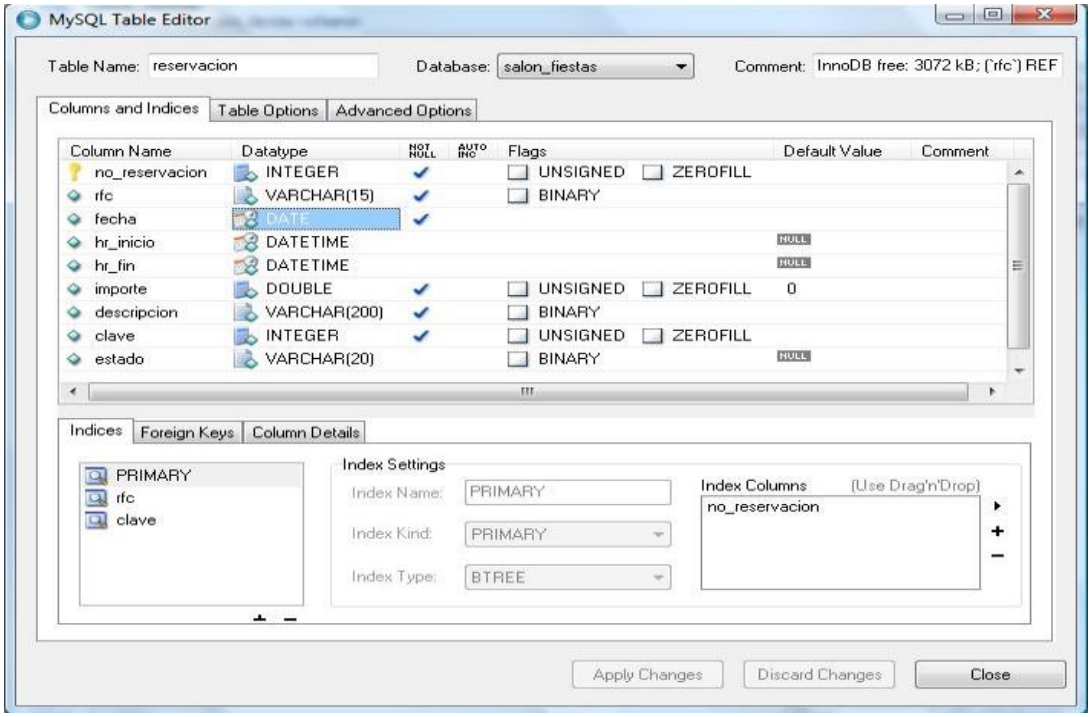


Figura 4.4 Tabla reservación

Definición de la tabla clasificación:

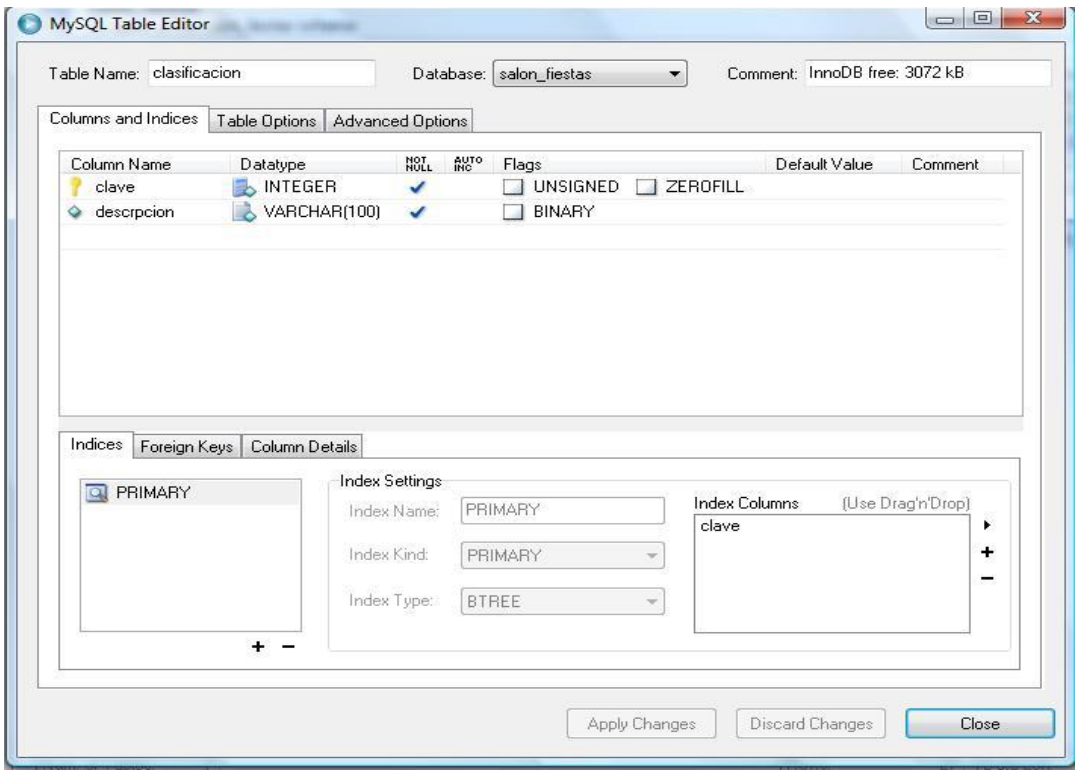


Figura 4.5 Tabla clasificación

Definición de la tabla contenido_reservacion:

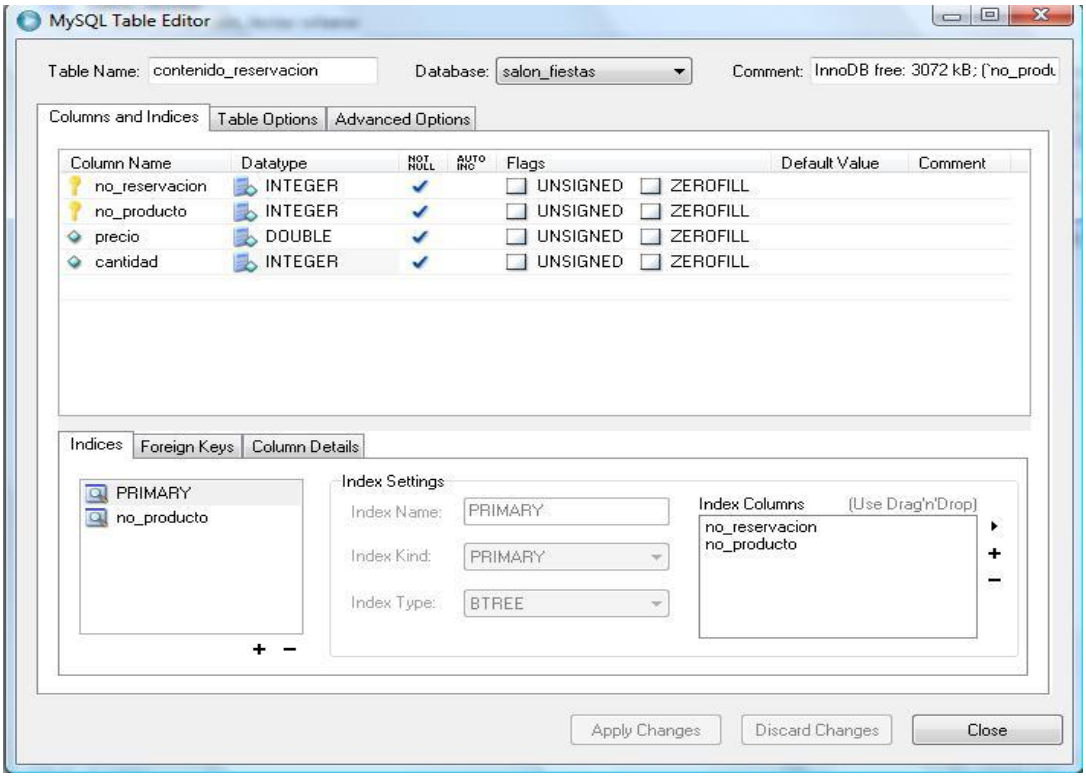


Figura 4.6 Tabla contenido_reservacion

Definición de la tabla

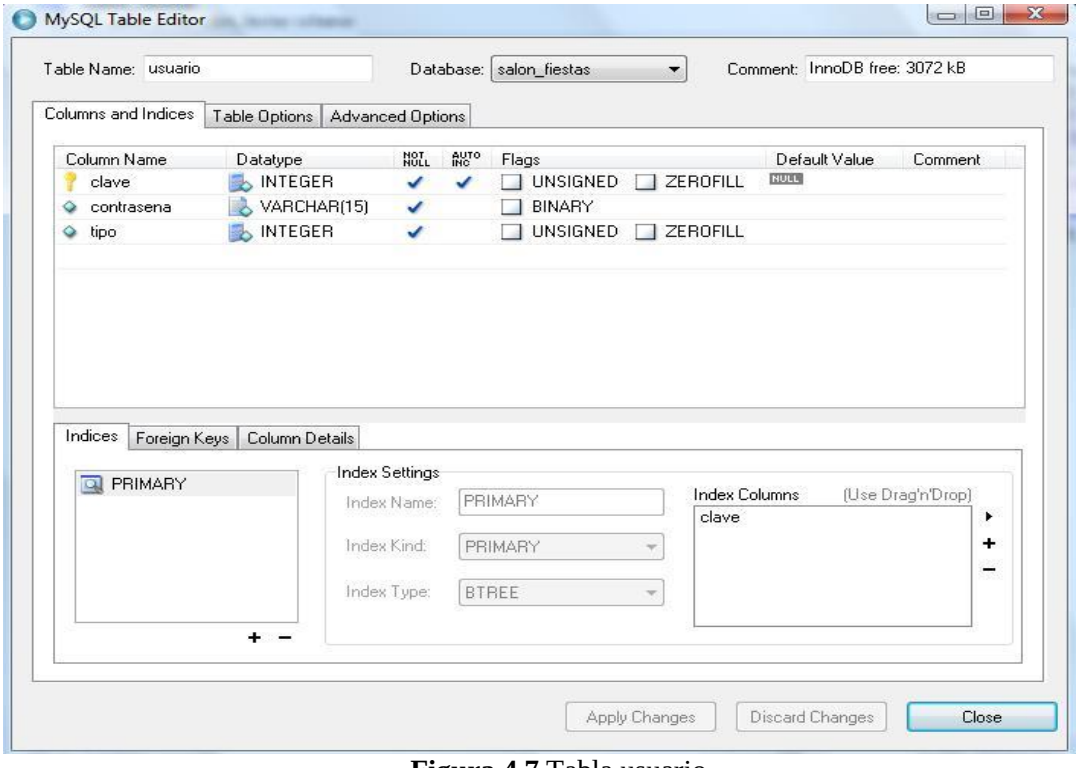


Figura 4.7 Tabla usuario

Definición de la tabla producto:

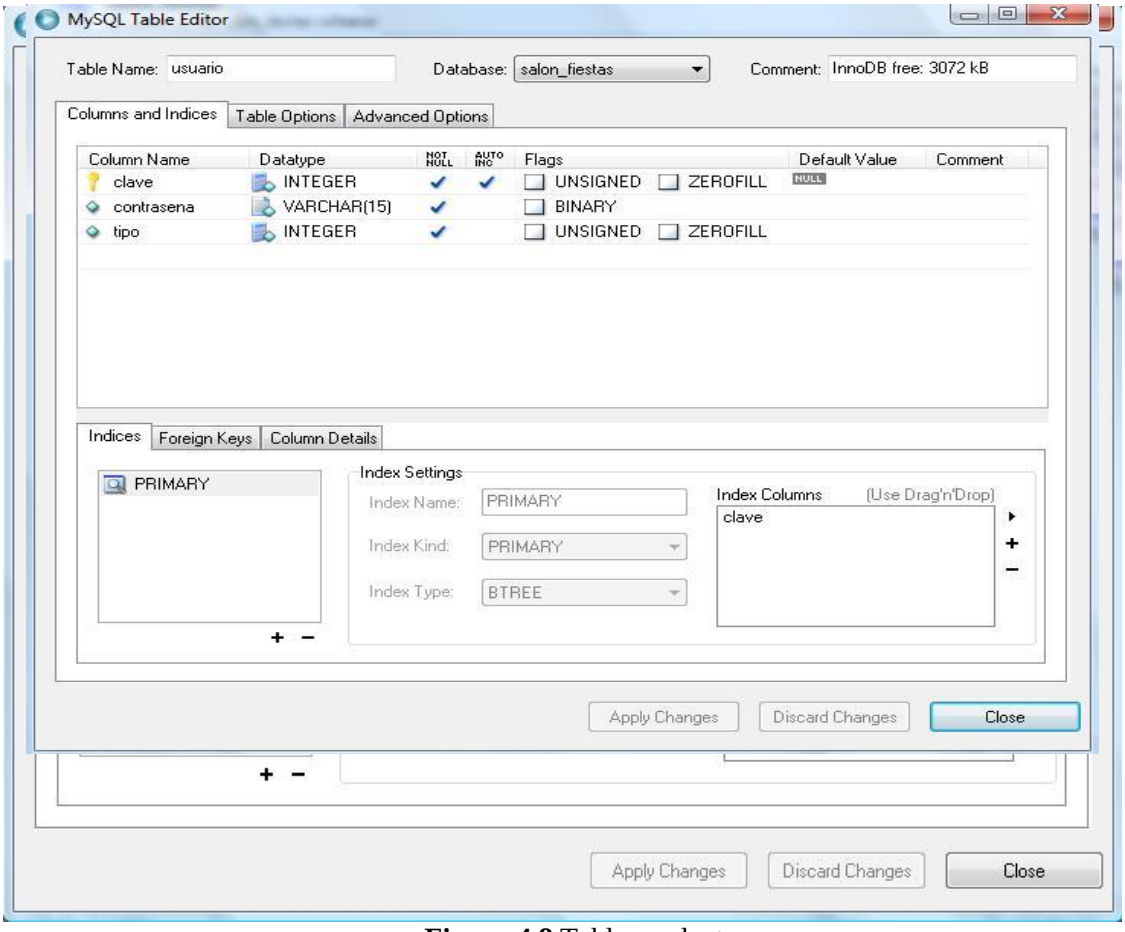


Figura 4.8 Tabla producto

4.5. Interfaces del Sistema

Al acceder al sistema se presenta la página de inicio del sistema de reservaciones (Figura 4.9), la cual describiremos a continuación:

Figura 4.9 Página de Inicio

1. En la parte superior izquierda se muestra la liga ***Home*** que permite regresar de forma directa a la página principal

2. ***Menú de navegación.*** se encuentra en la parte izquierda de la página actual y muestra las siguientes opciones:

- ✓ *Galería.* En esta opción se muestra la galería de las instalaciones y eventos del salón de fiestas.
- ✓ *Ubicación*. Muestra un mapa de la localización del lugar.
- ✓ *Sugerencias.* Por medio de esta opción el cliente puede consultar los menús más solicitados para los eventos.
- ✓ *Reservaciones.* Permite al cliente realizar la cotización y reservación de un evento.
- ✓ *Contáctanos.* En este apartado permite al cliente conocer los diferentes medios de comunicación con la empresa.
- ✓ *Administrador*. esta última sección permite administrar la información del sistema web. El acceso está restringido para usuarios de tipo administrador.

La implementación física del sistema de reservaciones utiliza la siguiente interfaz para acceder a las opciones del administrador.

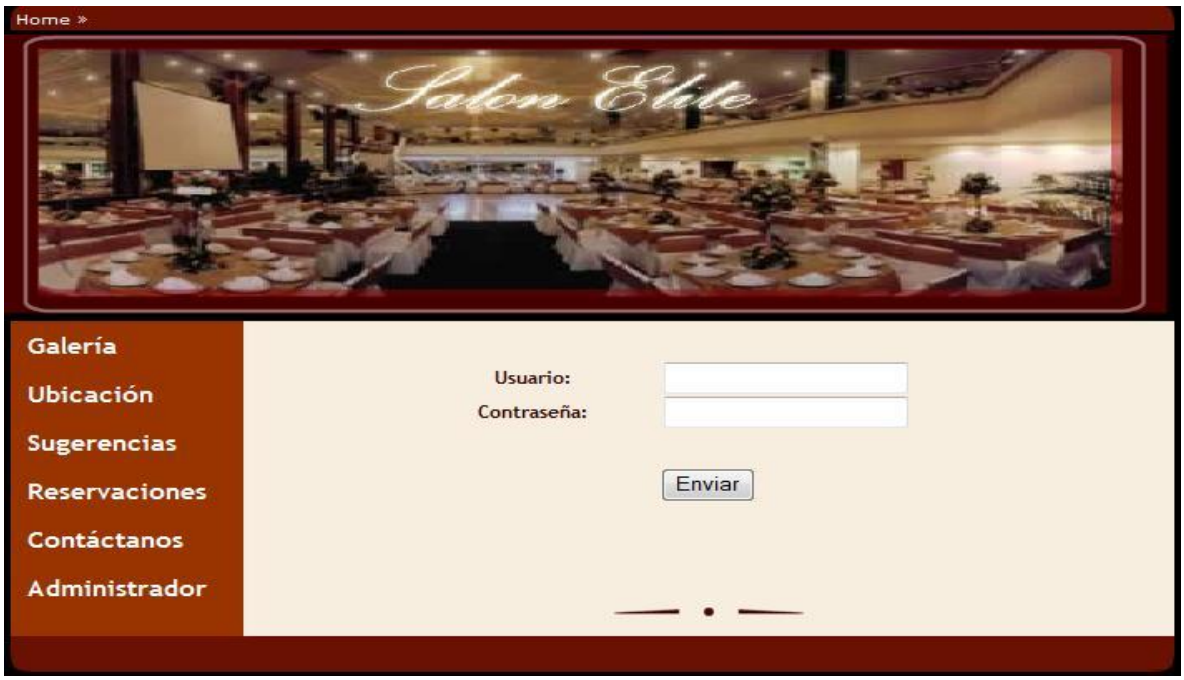


Figura 4.10 Acceso al sistema

En la figura 4.10 el usuario debe proporcionar el nombre de usuario y contraseña que le corresponde, si los datos son correctos se mostrará el menú de administrador.

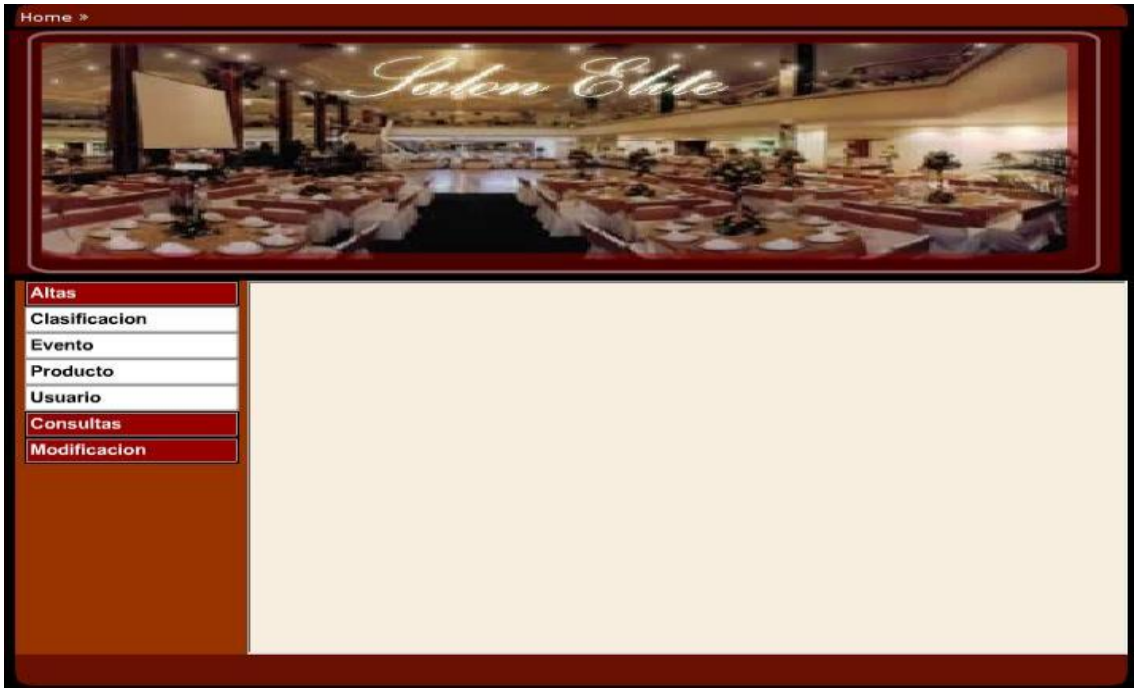


Figura 4.11 Menú administrador

Este menú (Figura 4.11) muestra tres secciones:

- ✓ Altas
- ✓ Consultas
- ✓ Modificaciones

En la sección *Altas* el usuario podrá agregar: Clasificaciones, eventos, productos y usuarios.
En la sección *Consultas* se obtienen los listados de la siguiente información: Clasificaciones, eventos, productos y reservaciones.

Desde la sección *Modificaciones* el usuario puede cambiar información de: Clasificaciones, eventos, productos y reservaciones.

Figura 4.12 Secciones del menú.

En seguida se muestran ejemplos de la ejecución de algunas de las operaciones anteriormente mencionadas. Empezaremos por mostrar los pasos que el usuario debe realizar para dar de alta un producto.

Figura 4.13 Alta de Producto.

Una vez que el usuario ha ingresado al menú administrador, elije de la sección *altas* la opción *producto*, enseguida se muestra un formulario con 3 campos (Fig. 4.13); el primero de ellos es la descripción que corresponde al nombre con el que se identificará el producto, el segundo campo contiene el precio de venta y el tercero designará la categoría a la que pertenece el producto. Después de que el usuario ha llenado la información solicitada en el formulario oprime el botón *agregar* y el sistema guarda los datos. Como alternativa el usuario podrá seleccionar la liga *ver productos* para verificar los productos existentes.

En el siguiente ejemplo hablaremos sobre cómo el usuario realiza la consulta de las reservaciones registradas en el sistema.

Cuando el usuario selecciona la opción *reservación* de la sección de *consultas*, en el área central aparece un listado con el resumen de las reservaciones, tanto activas como canceladas, que se han generado por los clientes (figura 4.14).

Figura 4.14 Consulta de Reservaciones.

Si el usuario da clic sobre el número de alguna reservación el sistema mostrará la información detallada relacionada con esa reservación. (Figura 4.15).

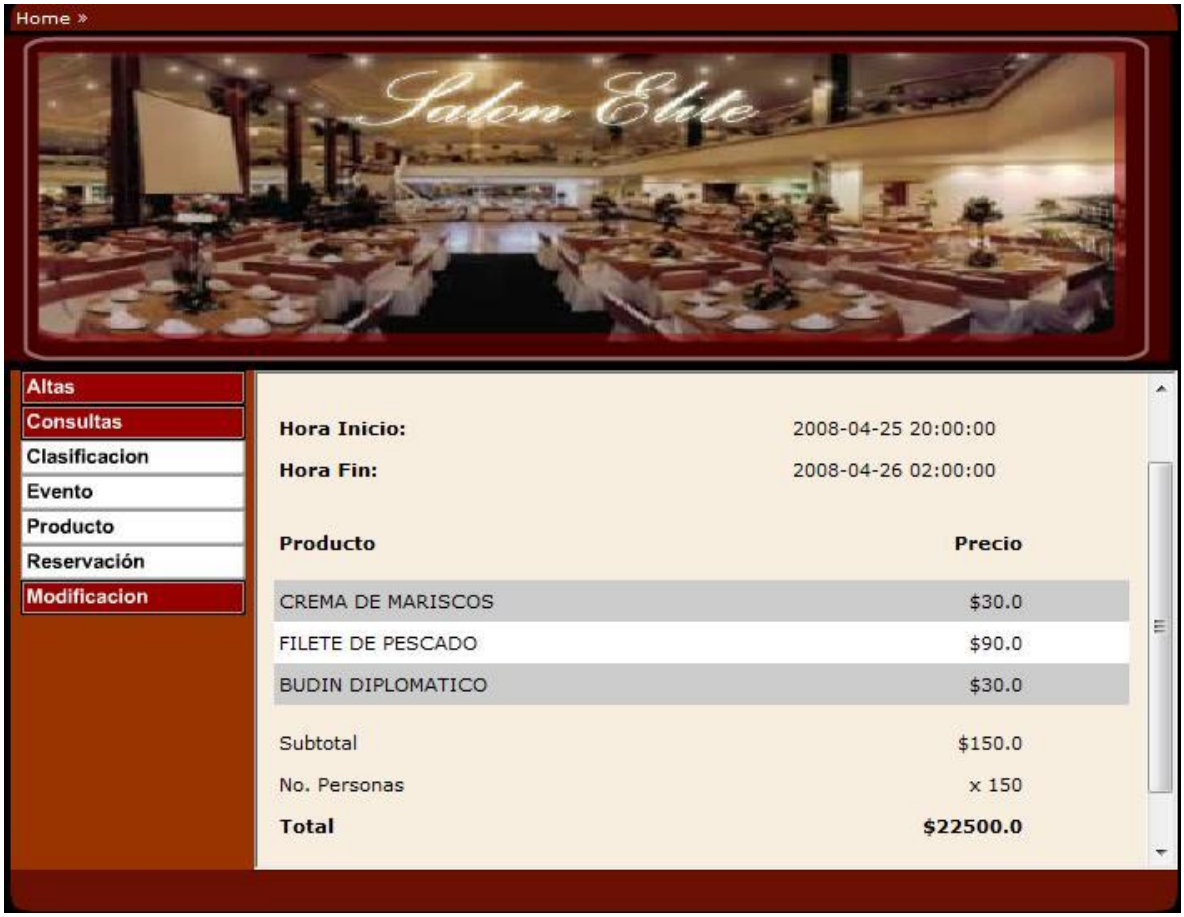


Figura 4.15 Detalle de Reservación.

Ahora se muestra un ejemplo del proceso que realiza el administrador para la modificación de un producto, observe que se puede realizar uno solo o varios a la vez.

Cuando el administrador selecciona la opción para modificar productos el sistema muestra un formulario que contiene todos los productos existentes en donde se permite modificar la descripción, precio o clasificación de cada uno de ellos (Figura 4.16). Una vez que el administrador ha realizado los cambios necesarios oprime el botón *modificar* para guardar los cambios.

Figura 4.16 Detalle de Reservación.

En este último ejemplo seguiremos el proceso para que un cliente pueda realizar una reservación a través del sistema de reservaciones.

El primer paso es seleccionar la opción reservación ubicada en el menú de navegación dentro de la página principal, de inmediato aparece una página que contiene un calendario, aquí el cliente deberá indicar la fecha, tipo y duración de su evento (Figura 4.17), al momento en que presiona el botón *siguiente* el sistema verifica las fechas seleccionadas permitiendo continuar si hay disponibilidad.

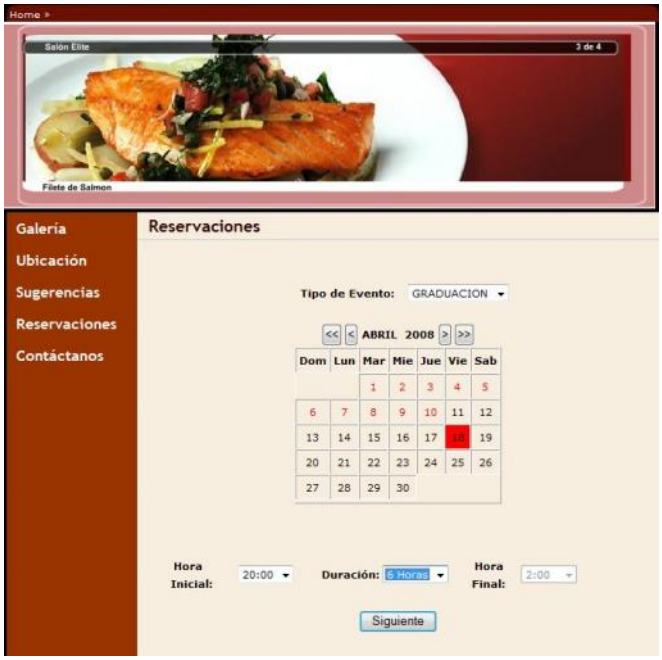


Figura 4.17 Selección de Fechas.

El siguiente paso permitirá al cliente agregar productos para personalizar su paquete así como indicar el número de personas que asistirán al evento (Figura 4.18), después es necesario dar clic en la liga *ver paquete* donde se le informará del costo total y se desplegará el resumen de su paquete (Figura 4.19).

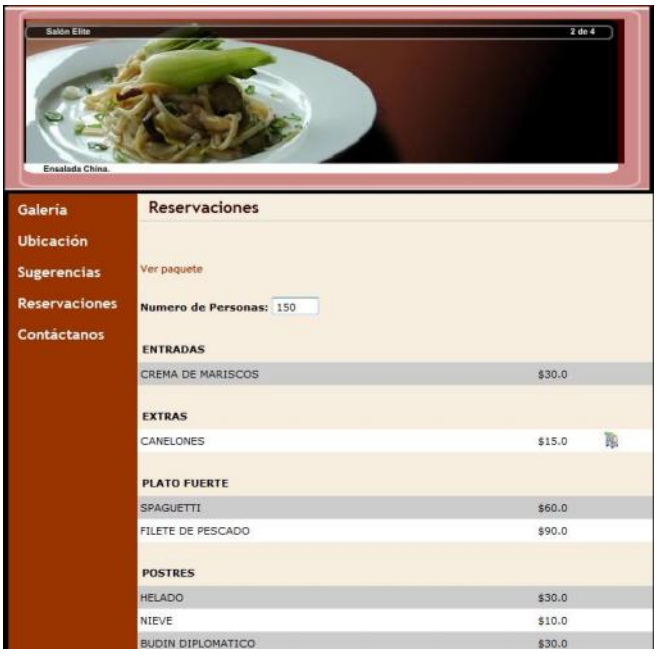


Figura 4.18 Selección de Platillos.

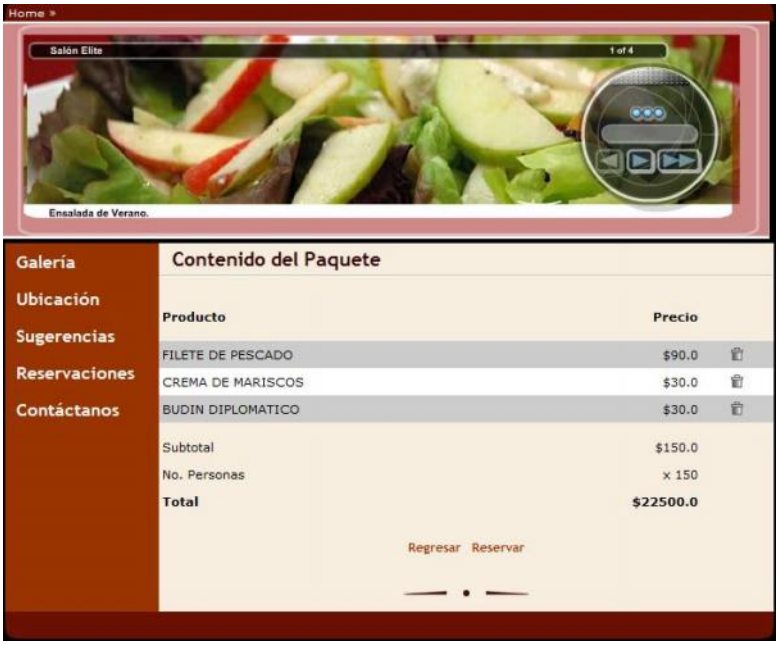


Figura 4.19 Contenido del paquete.

Si el cliente está conforme con su elección dará clic en *reservar* en donde se solicita que capture sus datos personales para poder concluir con la reservación (Figura 4.20).

Figura 4.20 Registro de Datos.

4.6. Pruebas del Sistema

En el desarrollo de cualquier sistema es necesario que exista una fase en donde se ejecuten varias pruebas para detectar deficiencias y corregirlas; estas pruebas también sirvieron para verificar que el sistema cumple con la especificación de los requerimientos. A continuación se muestran algunas de las pruebas que se realizaron en los módulos.

La Figura 4.21 muestra una prueba realizada en la página de inicio de sesión del administrador, en donde el usuario ha ingresado una contraseña inválida. Al hacer clic en el botón *enviar*, el sistema muestra una advertencia indicando que sus datos son incorrectos.

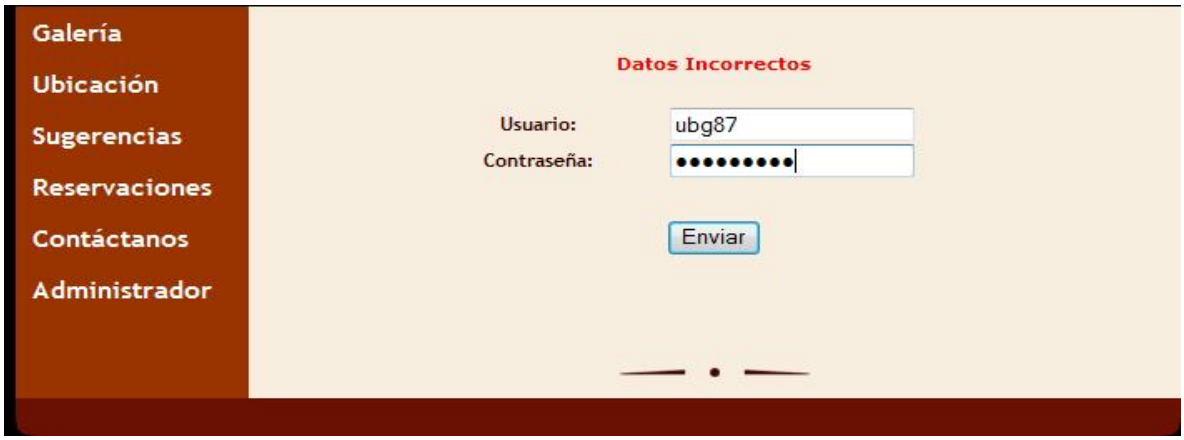


Figura 4.21 Prueba de inicio de sesión.

En la figura siguiente (fig.4.22) se intentó dar de alta un producto con una descripción que ya existe en la base de datos; al oprimir el botón *agregar* el sistema despliega un mensaje indicando que el producto ya estaba registrado.

Figura 4.22 Prueba para alta de productos.

Se realizó una consulta de productos, desplegándose un listado con todos los registros existentes en la base de datos, sin mostrar ninguna falla (Figura 4.23).

Figura 4.23 Prueba para consultar productos.

Se realizó una prueba para modificar eventos ingresando una descripción duplicada, al oprimir el botón *modificar* el sistema genera y muestra la notificación correspondiente y no se realiza la acción solicitada (Figura 4.24).

Figura 4.24 Prueba para modificar eventos.

Una de las validaciones más importantes se lleva a cabo cuando el usuario desea realizar una reservación y selecciona la fecha y duración de su evento. En la siguiente figura se muestra el caso en que las fechas indicadas por el usuario no están disponibles.

Figura 4.25 Prueba para reservar.

En el último paso de la reservación también se realiza una validación en el registro de datos del cliente mostrando un error si no se llenaron los campos requeridos o bien si el correo electrónico es incorrecto (Figura 4.26)

Figura 4.26 Prueba para registro de datos del cliente.

Este proyecto de tesis se enfocó a construir un sistema Web usando software libre y la notación UML en diversas fases del desarrollo del sistema.

El proyecto se orientó a cubrir las necesidades y requerimientos de usuarios reales cubiertos al 100%.

El paradigma de la ingeniería de software utilizando en este trabajo es el modelo en cascada que es aplicado durante el ciclo de vida del software, en la generación de código y pruebas. La elección del modelo en cascada, se decidió en base a las características del sistema a desarrollar, entre las que se pueden mencionar:

1. El sistema es de tamaño mediano y por tanto, es posible desarrollarlo en etapas secuenciales para tener un producto entregable en un tiempo definido y negociado con el usuario final.

2. Al inicio, se pudo definir de manera clara la mayoría de los requisitos, aunque hubo algunos ajustes de poca relevancia durante el proceso.

3. El modelo no provocó estados de bloqueo entre etapas.

Por medio del modelo relacional se definió la representación de los datos y el establecimiento de las relaciones de estos, lo cual nos conllevó a ofrecer un sistema que facilita, apoya y coadyuva a las actividades de control y manejo de la información.

Se ocupó MySQL como manejador de base de datos ya que es muy potente y pese a tratarse de un producto gratuito no tiene nada que envidiar de muchas bases de datos comerciales.

Para el desarrollo del sistema de reservaciones se utilizó JSP porque está basado en Java que es un lenguaje potente, escalable, robusto, bien estructurado, duradero y con un fácil mantenimiento.

Finalmente; en este trabajo se mostraron las ventajas de aplicar las Tecnologías de la Información en el desarrollo de aplicaciones reales.

5.2. Trabajo a futuro.

Una serie de posibles trabajos a futuro se listan a continuación:

- ✓ Que el cliente pueda consultar e imprimir sus reservaciones generadas
- ✓ Que el administrador pueda generar reportes:
  - productos más vendidos
  - frecuencia de clientes
  - eventos más solicitados
- ✓ Ampliar el catalogo de los productos
- ✓ Que el cliente pueda recibir su notificación por email

BIBLIOGRAFÍA.

[Allamaraju 00] Allamaraju, S. et al, (2000). *Professional Java Server Programming J2EE Edition*, Wrox Press.

[Arrington 01] Arrington CT. (2001). *Enterprise Java with UML*. Wiley Computer Publishing.

[De Miguel 01] De Miguel, A. et al (2001). *Diseño de bases de datos*, Alfaomega Ra-Ma.

[Froufe 02] Froufe, A. (2002). *JavaServer Pages Manual de usuario y tutorial*, Alfaomega Ra-Ma.

[Hunt 00] Hunt J. (2000). *The Unified Process for Practitioners*. Springer.

[Joyanes 98] Joyanes A. Luis. (1998), *Programación Orientada a Objetos*. Mc Graw Hill.

[Larman 99] Larman C. (1999). *UML y Patrones*. Prentice-Hall.

[Marty Hall 01] Marty Hall (2001). *Servlets y JavaServer Pages Guía Práctica*, Primera Edición, Prentice Hall.

[Silberschatz 02] Silberschatz, A. et al (2002). *Fundamentos de bases de datos* Cuarta Edición, Mc Graw Hill.

[Wutka 00] Wutka, M. (2000). *JavaServer Pages and Servlets*, Edición Especial, Que.

LIGAS EN INTERNET

[iPlanet] iPlanet E-Commerce Solutions. *Product Map.* http://www.iplanet.com/products/product_map/product_name_2_0a.html.

[Tomcat] The Jakarta Project. *Jakarta Tomcat.* http://jakarta.apache.org/tomcat.

[Sun] Sun Microsystems. *Java™ Servlet Technology: Implementations and Specifications.* http://java.sun.com/products/servlet/download.html.