UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Transferencia de Información entre una Base de Datos y una Aplicación de Comercio Electrónico mediante XML"

TESIS

Que para obtener el título de
LICENCIADA EN CIENCIAS
DE LA COMPUTACIÓN

PRESENTA
Edith Chavez Contreras

ASESOR
Dr. Mario Rossainz López

Puebla, Pue. Febrero 2008

Agradecimientos

Al Consejo de Ciencia y Tecnología del Estado de Puebla (CONCYTEP)

Por brindarme su cordialidad y apoyo mediante las Becas Tesis 2007.

A mis profesores y profesoras

Por aportarme sus conocimientos, consejos y experiencias.

Dedicatorias

A DIOS

Por darme la oportunidad de vivir, por la fortaleza que me brinda día con día para continuar este camino. Porque me ha guiado y cuidado en todo momento.

A mi papá José Concepción y a mi mamá Ofelia

Por su entrega, amor y dedicación; por su apoyo incondicional y su aliento en la consecución de todas mis metas.

A mis hermanas Yasmin, Pilar y Susana

Por la amistad y todas las vivencias compartidas.

A mi sobrino Ian Jonathan

Por compartir sus alegrías con migo.

A todos mis amigos y amigas

Por todos los gratos momentos que he vivido con ellos.

Resumen

En el presente proyecto de tesis se muestra un prototipo de aplicación Web de comercio electrónico, que permite intercambiar la información que procesa con su base de datos, utilizando archivos XML (eXtensible Markup Language, Lenguaje de Marcado Extensible) como una forma segura y estándar de llevar a cabo dicho intercambio de datos. Dentro de los modelos de negocios que existen en el comercio electrónico, se muestra una tienda virtual dedicada a la comercialización de música por Internet, Music e-Shop, dentro del paradigma de Negocio a Clientes.

Se muestra la realización del análisis y del diseño de dicha aplicación Web, con el propósito de describir cada una de las funcionalidades que la tienda virtual citada, en particular, debe cumplir, para con cada uno de los diferentes usuarios que harán uso de ella. Se utiliza el Lenguaje Unificado de Modelado (UML) en el marco de la Ingeniería Web Basada en UML (UWE). UWE propone la extensión en la semántica del UML puro hacia una nueva semántica, que representa de manera adecuada a cada uno de los elementos asociados al desarrollo de aplicaciones Web. Como resultado del trabajo del análisis y diseño, se obtiene una colección de diagramas en UML extendido, que constituye al modelo que describe la tienda virtual y que ha de utilizarse en la construcción de un prototipo de la aplicación Web referida.

Lo relevante de la aplicación es que una vez llevada a cabo la compra de un producto por parte del usuario y almacenados los datos de dicha compra en la base de datos, se utiliza esta información para emitir los comprobantes de compra correspondientes de la base de datos de la aplicación e-commerce. La escritura del archivo en formato XML con los datos de una compra, es llevada a cabo por un servlet. El archivo XML relacionado con su correspondiente Schema y hoja de estilos (XSL), es analizado por medio de un analizador sintáctico o parser, para generar una factura en formato HTML; que otro servlet devuelve a través del HTTP.

Cabe señalar que del proyecto interno de investigación de la FCC denominado: “Desarrollo de Aplicaciones Web usando arquitecturas de software JAVA con Modelado en UML extendido” parte el presente trabajo de tesis, para el que conté con el apoyo del CONCYTEP mediante las becas tesis 2007.

Índice General

Introducción

Motivación

Actualmente, el acceso de cualquier servicio Web a su base de datos, con el objetivo de procesar información, requiere de una serie de arquitecturas de software, que los programadores y usuarios tienen que saber, con el propósito de poder procesar su información a través de la Web, este último proceso depende del formato de base de datos que se utilice. Es por eso que, una posible solución es que la información sea procesada de forma estándar; es decir, sin necesidad de conocer específicamente el formato de la base de datos donde está almacenada la información. Además, incluye como interfaz el uso de documentos XML, con el fin de procesar la información de forma segura y estándar, independientemente de la arquitectura de software empleada en la aplicación Web que se trabaje.

En la actualidad, la tecnología .NET trabaja ya de esta manera, de forma transparente, al implementar servicios Web. Sin embargo, sabemos que esta tecnología no es abierta sino comercial, por lo que la propuesta consiste en ofrecer un procesamiento de información de la misma manera, pero usando arquitectura de software Java, la cual es abierta y no requiere mayores recursos económicos para poder utilizarla.

Objetivo

El objetivo de este proyecto es diseñar una aplicación de comercio electrónico, aplicando UWE como una metodología apropiada para el modelado de aplicaciones Web. Esta aplicación permite guardar en un archivo XML los datos extraídos de una base de datos. También, permite leer dichos archivos XML y analizarlos, así como mostrar los datos contenidos en aquellos archivos en formato HTML. Para así, de esta manera proporcionar recursos tecnológicos (específicamente de software) para la realización de comercio electrónico.

Organización del Documento

Este documento de tesis se encuentra organizado de la siguiente manera: En el primer capítulo se establece el marco teórico apropiado al problema. En dicho capítulo se hace referencia a conceptos relacionados con generalidades de las aplicaciones Web, el comercio electrónico y el Formato Estructurado de Texto (XML) en forma básica. También se presenta una introducción al análisis y diseño donde se describe las

actividades a tomarse en cuenta en el desarrollo del modelado de la Aplicación Web. En el capítulo 2 se inicia con el análisis, inicialmente con una propuesta, posteriormente con la especificación de requerimientos por medio de diagramas de casos de uso y finalmente con el comportamiento de dichos casos de uso. En el capítulo 3 se presenta el diseño del sistema, iniciando con el modelo lógico-conceptual; continuando con la presentación de los diagramas de clases estereotipadas que constituyen al modelo de navegación. Posteriormente se presentan los diagramas de secuencia que describen interacciones temporales entre elementos descritos en el modelo de navegación. Para finalizar este capítulo se presenta una colección de escenarios Web descritos mediante diagramas de estados. En el capítulo 4 se aporta un panorama de la implementación adecuada para la Tienda Virtual siguiendo el modelado de la aplicación y empleando XML. Al final del documento se presentan las conclusiones del trabajo realizado, dos apéndices y la bibliografía empleada en el presente proyecto. El apéndice A se refiere al diseño de la base de datos y el apéndice B a la configuración de la plataforma de desarrollo de aplicaciones Web con Java.

Capítulo 1

Marco Teórico

En este capítulo se hace referencia a conceptos relacionados con generalidades de las aplicaciones Web, el comercio electrónico y el Formato Estructurado de Texto (XML) en forma básica. También se presenta una introducción al análisis y diseño donde se describe las actividades a tomarse en cuenta en el desarrollo del modelado de las aplicaciones Web.

1.1 Aplicaciones Web

El concepto de Internet esta muy difundido y representa un medio de comunicación abierto donde cualquier persona, grupo, organismo, corporación o gobierno pueden publicar sus ideas, productos, conocimientos, promociones, entre otros. Además de que los navegadores pueden animar imágenes, reproducir flujos de audio y video y una gran variedad de información, dando como resultado aplicaciones más atractivas.

En este apartado se presentan los fundamentos de las aplicaciones Web con la finalidad de tener una pauta introductoria para el resto del documento.

1.1.1 Internet

Internet (International Network of Computers) es una red de redes informáticas distribuidas por todo el mundo que intercambian información entre si mediante la familia de protocolos TCP/IP [Rossainz06]. El acceso a Internet es cada vez más atractivo, económico y cotidiano, que permite una comunicación global. Debido a esto, el uso de Internet se ha expandido para formar parte de los negocios, además de la utilización académica e institucional que tuvo originalmente.

1.1.2 Los servicios en Internet

Entre los servicios más comunes en Internet figuran los siguientes: conectarse a una terminal desde otro lugar (telnet); transferir archivos entre una computadora local y una computadora remota (protocolo de transferencia de ficheros, o FTP) y leer e interpretar

archivos de ordenadores remotos (gopher). Internet permite también intercambiar mensajes de correo electrónico (e-mail); acceso a grupos de noticias y foros de debate (news), y conversaciones en tiempo real (chat, IRC), entre otros servicios.

La WWW (World Wide Web) o, de forma más simple, la Web, es un sistema hipermedia interactivo que permite conectarse a grandes cantidades de información en Internet. Un sistema hipermedia esta compuesto por páginas que contienen texto, imágenes, sonido, video y enlaces a otras páginas distribuidas por todo el mundo. La Web es un servicio que permite acceder a numerosas prestaciones y funciones, así como a múltiples servicios, programas, negocios, etc.

1.1.3 Cimientos de la Web

Las dos columnas fundamentales de la Web son: el protocolo HTTP y el lenguaje HTML. El primero permite una implementación simple y sencilla de un sistema de comunicaciones que nos permite enviar cualquier tipo de ficheros de una forma fácil, simplificando el funcionamiento del servidor y permitiendo que servidores poco potentes atiendan miles de peticiones y reduzcan los costes de despliegue. El segundo nos aporta un mecanismo de composición de páginas enlazadas simple y fácil, altamente eficiente y de uso muy simple. [Mateu04]

1.1.3.1 El protocolo HTTP

El protocolo HTTP (HyperText Tranfer Protocol) es el protocolo base de la WWW. Se trata de un protocolo simple, orientado a conexión y sin estado. La razón de que esté orientado a conexión es que emplea para su funcionamiento un protocolo de comunicaciones (TCP, transport control protocol) de modo conectado, un protocolo que establece un canal de comunicaciones de extremo a extremo (entre el cliente y el servidor) por el que pasa el flujo de bytes que constituyen los datos que hay que transferir, en contraposición a los protocolos de datagrama o no orientados a conexión que dividen los datos en pequeños paquetes (datagramas) y los envían, pudiendo llegar por vías diferentes del servidor al cliente. El protocolo no mantiene estado, es decir, cada transferencia de datos es una conexión independiente de la anterior, sin relación alguna entre ellas, hasta el punto de que para transferir una página web tenemos que enviar el código HTML del texto, así como las imágenes que la componen, pues en la

especificación inicial de HTTP, la 1.0, se abrían y usaban tantas conexiones como componentes tenía la página, trasfiriéndose por cada conexión un componente (el texto de la página o cada una de las imágenes).

De manera esquemática, el funcionamiento de HTTP es el siguiente: el cliente establece una conexión TCP hacia el servidor, hacia el puerto HTTP (o el indicado en la dirección de conexión), envía un comando HTTP de petición de un recurso (junto con algunas cabeceras informativas) y por la misma conexión el servidor responde con los datos solicitados y con algunas cabeceras informativas. [Mateu04]

1.1.3.2 El lenguaje HTML

HTML (HyperText Mark-up Language) es un lenguaje de marcas comúnmente utilizado para la publicación de hipertexto en la Web y desarrollado con el objetivo de que cualquier persona o tipo de dispositivo pueda acceder a la información en la Web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento [W3C], determinando la apariencia y funcionalidad de una página Web.

Las etiquetas de HTML se escriben entre paréntesis angulares (< >). Regularmente existe una etiqueta de apertura y su correspondiente etiqueta de cierre, diferenciándose esta ultima por tener después del signo < una diagonal (/).

La primera línea de un archivo HTML debe contener a la etiqueta <HTML> y la ultima línea del documento debe contener la etiqueta de cierre </HTML>. La estructura básica de un documento HTML consiste en dos secciones. Una sección de encabezado marcada por la etiqueta <HEAD> texto </HEAD>, la cual puede contener información que describa el contenido de la pagina mediante el uso de varios elementos. Y una sección del cuerpo del documento encerrada por la etiqueta <BODY> contenido </BODY>, la cual contiene el texto y los elementos a publicar en la pagina con las etiquetas necesarias para darles formato.

Los documentos HTML son archivos de texto ASCII que pueden ser creados en cualquier editor de texto o en procesadores de palabras los cuales deben de guardarse con la extensión .html o .htm. Incluso existe software especializado para este propósito. El archivo de texto mostrado en la Figura 1.1 muestra la estructura básica de un documento HTML y el resultado visto desde el navegador.

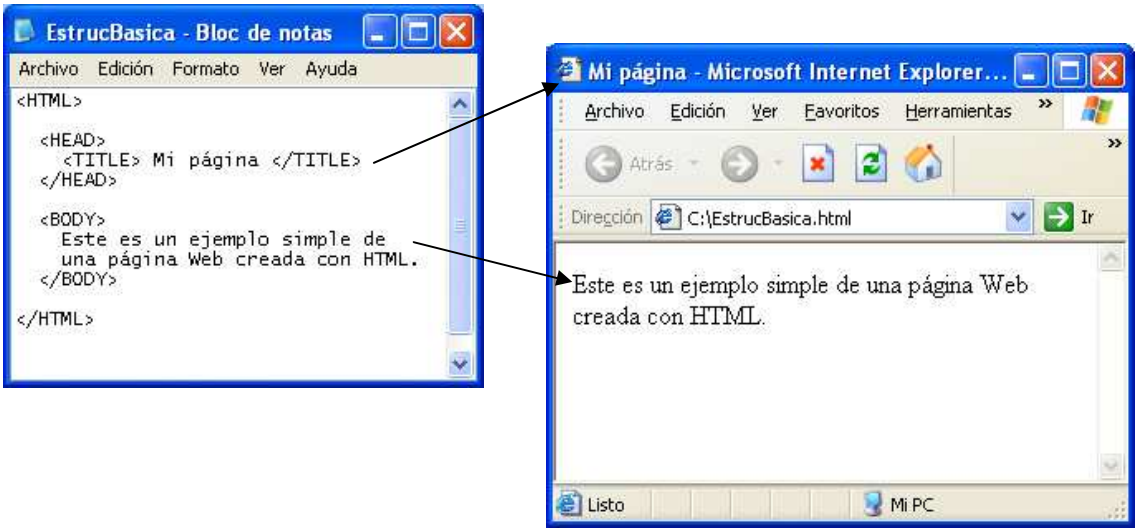

Figura 1.1: Ejemplo de un documento HTML.

Para dar formato y mejorar la presentación de texto se tienen etiquetas para establecer encabezados con seis diferentes tamaños de caracteres, centrados, separación de párrafos, saltos de línea, atributos como negritas y cursivas, listas y tablas. El uso de ligas o vínculos es una de las principales características en los documentos HTML, los cuales permiten la navegación entre páginas tan solo con un clic del mouse. Además de la creación de paginas Web atractivas mediante la inclusión de imágenes, videos y sonido.

1.2 Comercio Electrónico

En un tiempo notablemente corto, el Internet ha pasado de ser un recurso opcional de comunicación a ser un medio vital e imprescindible para la vida diaria de la sociedad, incluyendo los negocios. A la par de la Internet, la Web evoluciona a pasos agigantados, es por esto que la manera de dirigir y administrar negocios de manera exitosa debe expandir sus horizontes a fin de adaptarse y obtener los mayores beneficios de la Internet. Uno de los vehículos principales del cambio es el Comercio Electrónico, el cual permite a las empresas ampliar su campo de acción, dar mejor respuesta a las necesidades de los Clientes, y lo más importante, ofertar sus productos en un mercado global.

En este apartado se describen conceptos, definiciones y generalidades introductorias para el comercio electrónico que, en su conjunto, establecen un marco de referencia para el resto del documento [Ocaña06].

1.2.1 Definición de Comercio Electrónico

Algunos estudiosos opinan que, en cierta forma, el comercio electrónico surgió aún antes de Internet, mediante transacciones comerciales por teléfono y fax, pero el desarrollo de la Web motivó que alcanzara mayor auge, por su masividad y rapidez de operación. La acepción más general que se pude hacer de comercio electrónico es la de "acercar el comprador al fabricante por medios electrónicos", lo cual implica eliminación de intermediarios, reducción de costos y una filosofía diferente en la forma de comprar y vender, y lo que es más importante, de obtener información para esas gestiones.

El comercio electrónico consiste en llevar a cabo cualquier actividad de negocio relacionada con la comercialización de productos o servicios a través de redes informáticas, aplicando para ello tecnologías de información dirigidas a realizar dichas transacciones comerciales [Ocaña06].

1.2.2 Actores y Transacciones en Sistemas de Comercio Electrónico

Al igual que en el comercio presencial, en el comercio electrónico intervienen diversos actores que deben interactuar entre si para llevar a cabo las transacciones de compraventa. Los actores mínimos de un sistema de comercio electrónico:

- Comprador: Es todo aquel que adquiere un producto o servicio.
- Vendedor: Aquel que promociona y vende un producto o servicio.
- Entidades y Servicios Financieros: Validan y autorizan pagos y realizan los movimientos de dinero entre comprador y comerciante.

Cabe comentar que las entidades y servicios financieros pueden ser diversos aún en una misma transacción, por ejemplo: el banco del comprador, el banco del Vendedor, la pasarela entre bancos, marca de plásticos (débito o crédito), el broker para micro pagos

(corredor o agente que actúa como intermediario entre un comprador y un Vendedor, usualmente cobrando una comisión, por ejemplo Paypal), por citar algunos.

Todos estos actores en una transacción han de estar comunicados mediante la red (Internet). Dependiendo de la forma de pago, sistemas de seguridad y otros factores de logística pueden entrar en acción en la transacción otros actores como lo pueden ser Empresas de Paquetería y transporte para el envío de los productos.

Una transacción electrónica siempre se realiza con al menos 4 fases:

1. El comprador obtiene los datos del producto o servicio del vendedor.
2. El comprador paga y se solicita la autorización de dicho pago.
3. Se confirma la autorización y se paga al vendedor (a veces el pago se realiza después de la cuarta fase).
4. Se entrega el producto o servicio mediante un transporte físico o por la red.

Un sistema de comercio electrónico debe de ser capaz de automatizar las 3 primeras fases, debe presentar, promocionar y facilitar la adquisición del bien, gestionar la autorización del pago y confirmado el pago debe iniciar el proceso de entrega del bien [Ocaña06].

1.2.3 Tipos de Comercio Electrónico

Con la evolución de las tecnologías de información y las crecientes necesidades de las organizaciones han surgido diferentes maneras de hacer comercio electrónico. A estas formas se le han llamado tipos de Comercio electrónico (algunos autores le llaman clasificación de Comercio electrónico) y se distinguen esencialmente por el tipo de actores (entiéndase usuarios) que participan en él. De entre los más importantes podemos mencionar los siguientes:

- Negocio a Clientes (Business to Customers): Es el comercio electrónico que realizan las empresas con los particulares. En este género se incluyen todos aquellos negocios que utilizan Internet para comercializar cualquier tipo de bienes y/o servicios al público en general, desde libros, juguetes, equipos electrónicos, entre otros, de manera individual. Este tipo de transacción permite a las empresas mostrar su oferta en línea y mostrar sus productos a través de catálogos. En este tipo de comercio electrónico el segmento de clientes es toda la red. Los expertos aseguran

que la gran ventaja de esta forma de venta en línea es que no se necesitan grandes cantidades de inventario físico sino solo rápidas soluciones de distribución.

- Negocio a Negocio (Business to Business): Es el comercio electrónico que realizan las empresas con otras empresas. Se refiere a las transacciones comerciales realizadas entre empresas exclusivamente, generalmente bajo un esquema de distribuidores y proveedores. Involucra a los procesos contractuales electrónicos en los que participan dos o más empresas, bien a través de las personas físicas que las representan o bien a través de sus propios sistemas informáticos. Esta rama es muy amplia y se desarrolla utilizando el Internet como plataforma múltiple y neutral de comunicación.
- Clientes a Clientes (Customer to Customer): Es el comercio electrónico que se realiza entre particulares. Se refiere a las transacciones comerciales privadas entre personas individuales que pueden tener lugar mediante simplemente el intercambio de correos electrónicos, el uso de tecnologías P2P (Peer to Peer, Punto a Punto) o bien utilizando complejas aplicaciones Web especializadas como Ebay por citar alguna.
- Negocio a Empleado (Business to Employe): Se refiere a la relación comercial que se establece entre una empresa y sus propios empleados. Esta tendencia surgió debido a problemas de seguridad que hicieron que las empresas utilizaran las tecnologías ya no en Internet sino hacia adentro a través de un Intranet para tener mayor control y seguridad en sus transacciones.

Estos 4 esquemas son los que predominan en la red. Adicionalmente, se identifica otra más llamada Negocio a Gobierno (Business to Government), llamado también “gobierno electrónico”, que permite atender al ciudadano a través de la red para trámites aduanales, certificaciones, seguridad social, pagos, cobros y otros servicios oficiales [Ocaña06].

1.3 Formato Estructurado de Texto: XML

La Word Wide Web generó un nuevo estándar llamado XML que nace de las especificaciones de SGML (Estándar Generalized Markup Language) desarrollado desde 1996 , dándose a conocer la versión XML 1.0 en 1998 siendo definido como un “un sistema para definir, validar y compartir formatos de documentos en la Web”.

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el proceso de intercambio, estructuración y envío de datos en la Web. Es un lenguaje similar a HTML pero su función primordial es describir datos y no mostrarlos como es el caso de HTML. Con XML es posible estructurar los datos utilizando para ello etiquetas no predefinidas con el fin de delimitar dichos datos y simultáneamente beneficiar la interoperabilidad de los mismos. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones [W3C].

XML es un metalenguaje, es decir, un lenguaje para la definición de lenguajes de marcado. XML define la sintaxis y los requisitos que deben cumplir los lenguajes de marcado que especifica. XML en definitiva organiza y permite nuevas formas de incluir información extra, a través de marcas, a nuestros textos, esta información adicional permite realizar sobre estos textos enriquecidos un gran numero de tareas informáticas y limitado únicamente por la información añadida, por la necesidad y por la imaginación Algunas de estas podrían ser: búsquedas mas precisas, personalización de la información, generación automática de informes, filtrados, etc. [Gutiérrez01].

1.3.1 Estructura XML

Un documento XML esta formado por una estructura lógica y una física, físicamente el documento esta compuesto por unidades llamadas entidades. Lógicamente, el documento esta formado de declaraciones, elementos, comentarios e instrucciones de procesamiento, los cuales correspondientemente están indicados con una marca explícita. Estas dos estructuras deben encajar adecuadamente para dar lugar a los documentos XML que pueden ser bien formados o documentos válidos [Charles99].

El documento XML se divide en dos partes, el encabezado y el cuerpo. La primera donde se realizan todas las declaraciones formales del documento, y la segunda utiliza una etiqueta raíz debajo de la cual se encuentra el resto de etiquetas, atributos y demás elementos que forman el documento XML.

Ejemplo de documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<libro>
  <titulo></titulo>
  <capitulo>
    <titulo></titulo>
```

```
    <seccion>
      <titulo></titulo>
    </seccion>
  </capitulo>
</libro>
```

1.3.1.1 Definición de Tipo de Datos (DTD)

Los DTD sirven para definir un lenguaje a partir de las reglas que marca SGML. Así han nacido lenguajes como HTML o DocBook, y muchos mas, dirigidos a distintas áreas del conocimiento. En los DTD, como su nombre indica, se definen los objetos que van a formar parte de los documentos XML que se construyan siguiendo ese DTD en concreto [Vázquez02].

Cuando se utiliza una etiqueta, los DTD definen entre otras cosas: que tan extenso puede ser su valor, el tipo de carácter (UTF-8, UTF-16...), reglas que deben cumplirse en la información (ser parte de otra etiqueta, valores específicos...), referencias a otros DTD. Es decir los DTD definen como serán empleados e interpretados los elementos de un documento XML.

1.3.1.2 Schemas

Los Schemas han surgido como una alternativa a los DTD utilizados para validar información en XML, pero de una forma mucho más precisa. Los Schemas utilizan la sintaxis XML para ser construidos, mientras que los DTD se encuentran en EBNF (Extended Backus Naur Form). Los Schemas permiten definir la estructura, contenido y semántica de documentos XML.

El propósito de un Schema es definir y describir una clase de documentos XML usando estas construcciones para restringir y documentar el significado, uso y relaciones de las partes constituidas: tipo de datos, elementos y su contenido, atributos y sus valores, entidades y su contenido, y anotaciones. Los esquemas documentan su propio significado, uso y función [W3C].

1.3.2 Lenguaje Extensible de Hojas de Estilo (XSL)

XSL es un lenguaje para crear hojas de estilo a través de las cuales será posible mostrar el contenido estructurado de un documento con un formato determinado. Por lo

tanto el objetivo principal de XSL es mostrar cómo debería estar estructurado el contenido, cómo debería ser diseñado el contenido de origen y cómo debería ser paginado en un medio de presentación como puede ser una ventana de un navegador Web o un dispositivo de mano, o un conjunto de páginas de un catálogo, informe o libro. XSL consiste en dos partes: un lenguaje de transformación de documentos XML y un vocabulario XML para especificar semánticas para el formato (objetos de formato).

Transformaciones del Lenguaje de Hojas de Estilo Extensible (XSLT). Es un lenguaje que permite la transformación de la estructura de un documento XML en otro documento XML con estructura diferente [W3C].

Además de XSL se cuenta con las Hojas de Estilo en Cascada (CSS), el cual es un mecanismo para dar estilo a documentos HTML y XML, que consiste en reglas simples a través de las cuales se establece cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir [W3C].

Como se puede observar en la tabla 1.1, tanto CSS como XSL se utilizan para dar estilo a XML. Por otro lado, a diferencia de CSS, XSL es exclusivo para XML, utiliza la sintaxis XML y consta de un Lenguaje de Transformación. Por lo tanto, es más recomendable utilizar XSL.

	CSS	**XSL**
¿Puede usarse con HTML?	Sí	No
¿Puede usarse con XML?	Sí	Sí
¿Lenguaje de transformación?	No	Sí
Sintaxis	CSS	XML

Tabla 1.1: Comparaciones entre XSL y CSS.

1.3.3 DOM y SAX

Los documentos XML se analizan por medio de procesadores, los cuales son módulos de software que permiten leer los documentos XML y proporcionar programas de aplicación con acceso a su contenido y estructura. Para procesar un documento XML después de acceder a su estructura interna, es necesario utilizar una API (Interfaz de Programación de Aplicaciones). A continuación mencionaremos dos API que se utilizan en procesadores XML:

1.3.3.1 DOM

DOM (Document Object Model) Es una especificación desarrollada por el "World Wide Web Consortium" que define como procesar ("parser") documentos en XML. DOM genera un árbol jerárquico en memoria del documento o información en XML. Dicho árbol permite que a través del "parser" (Xerces o algun otro) sea manipulada la información, las ventajas serian las siguientes:

- Puede ser agregado un nodo (Información) en cualquier punto del árbol.
- Puede ser eliminada información de un nodo en cualquier punto del árbol.

1.3.3.2 SAX

SAX (Simple API for XML) procesa la información por eventos. SAX procesa la información en XML conforme esta sea presentada (evento por evento), efectivamente manipulando cada elemento a un determinado tiempo, sin incurrir en uso excesivo de memoria. Por lo tanto se pueden notar las siguientes características:

- Es ideal para manipular archivos de gran tamaño, ya que no ocupa generar un árbol en memoria como es requerido en DOM.
- Es más rápido y sencillo que utilizar DOM.
- Debido a que SAX funciona por eventos no es posible manipular información una vez procesada.

1.4 La Ingeniería Web Basada en UML

En esta sección se presentan los términos y conceptos sobre los cuales se fundamentan los modelos descritos en los siguientes 2 capítulos. El punto a tratar es la Ingeniería Web basada en UML que se empleara como guía de referencia en el análisis y diseño de Music e-Shop.

La Ingeniería Web basada en UML (UWE, UML-based Web Engineering), es un proceso del desarrollo para aplicaciones Web enfocado sobre el diseño sistemático, la personalización y la generación semiautomática de escenarios que guíen el proceso de desarrollo de una aplicación Web. UWE describe una metodología de diseño sistemática, basada en las técnicas, la notación y los mecanismos de extensión de UML

como los estereotipos, todo esto debido a que las aplicaciones Web tienen características especiales a diferencia de los sistemas de software tradicionales. Para satisfacer estas características, UWE define vistas especiales representadas gráficamente por diagramas en UML y utiliza y define estereotipos para el modelado de aplicaciones Web. A esto se le conoce como Perfil UML [RossOcaCCV06].

1.4.1 Actividades de Modelado de una Aplicación Web

Los modelos y diagramas necesarios para lograr representar de manera satisfactoria los elementos arquitectónicamente significativos de una aplicación Web, según [RossOcaUWE06], se describen a continuación:

- Especificación de requerimientos.- El modelo de casos de uso de UML se puede utilizar para describir los requisitos funcionales de una aplicación en términos de los casos de uso. Un caso de uso en UML es una unidad coherente de la funcionalidad proporcionada por la aplicación que obra recíprocamente con unos o más actores de la aplicación. Describe una parte del comportamiento de la aplicación sin revelar la estructura interna. De esta manera, los requisitos para una aplicación Web se pueden especificar con un modelo de casos de uso.
- Modelo Lógico-Conceptual.- Un diagrama de clases en UML se utiliza para representar gráficamente un modelo conceptual como visión estática que demuestre una colección de los elementos estáticos del dominio. UWE apunta a construir un modelo conceptual de una aplicación Web, la cual procura no hacer caso en la medida de lo posible de cuestiones relacionadas con la navegación, y de los aspectos de interacción de la aplicación Web. Estos aspectos se posponen a los pasos navegacionales y de presentación del diseño. La construcción de este modelo lógico-conceptual se debe llevar a cabo de acuerdo con los casos de uso que se definen en la especificación de requerimientos. El modelo conceptual incluye los objetos implicados en las actividades típicas que los usuarios realizarán en la aplicación Web, es decir, los objetos que son relevantes para la realización de una actividad o que son el resultado de una de ellas.
- Modelo de Navegación.- El modelo de navegación de una aplicación Web comprende la especificación de qué objetos pueden ser visitados mediante la navegación a través de la aplicación Web y las asociaciones entre ellos. Los

modelos de la navegación son representados por los diagramas de clases estereotipadas. Este modelo se destaca en el marco de UWE como el más importante, pues con él se pueden representar elementos estáticos, a la vez que se pueden incorporar lineamientos semánticos de referencia para las funcionalidades dinámicas de una aplicación Web.

- Interacción Temporal.- Un diagrama de secuencia de UML demuestra la interacción de elementos dispuesta en orden temporal. Presenta los objetos que participan en la interacción y la secuencia de los mensajes enviados entre ellos. UWE propone el uso de los diagramas de secuencia para representar los aspectos dinámicos de la navegación, es decir, las secuencias describen la realización de los casos de uso. De esta manera, los diagramas de secuencia proveen una representación funcional centrada en el tiempo del modelo de navegación.
- Escenarios Web.- Un diagrama de estados de UML denota una secuencia de los estados que un objeto puede adquirir durante su vida, junto con acciones responsivas, disparando eventos y las condiciones asociadas para indicar transiciones. UWE da otro sentido a los diagramas de estados del UML puro ya que los utiliza para visualizar escenarios de navegación. Estos diagramas permiten detallar la parte dinámica del modelo de navegación, especificando los eventos que disparan las situaciones, definen condiciones y explícitamente incluyen las acciones que son realizadas. Junto con el modelo de interacción temporal, los escenarios Web proveen la representación funcional dinámica del modelo de navegación.

Con la construcción de estos modelos, UWE permite la descripción completa de las funcionalidades que debe realizar una aplicación Web. La especificación de requerimientos es la fase que comprende la parte de análisis y las etapas restantes, modelos Lógico-Conceptual, de Navegación, Interacción Temporal y Escenarios Web, comprenden la parte del diseño del sistema. En los dos siguientes capítulos se muestran los diagramas y modelos en el proceso de análisis y diseño de Music e-Shop.

Capítulo 2

Análisis: Especificación de Requerimientos

En este capítulo se presenta la especificación de requerimientos para una tienda virtual como un tipo de aplicación de comercio electrónico. Los requerimientos de cada uno de los usuarios de la tienda virtual se describen mediante diagramas de Casos de Uso en UML. En el primer apartado se muestra la descripción de un prototipo de una tienda virtual que proporciona los requisitos necesarios para su análisis. En el segundo se presentan los diagramas de casos de uso de la tienda virtual en base a su delineación propuesta. Y en el tercero se demuestra el comportamiento de los casos de uso que son referidos en el segundo apartado.

2.1 Prototipo de una Tienda Virtual

Dentro de los modelos de negocios que existen en el comercio electrónico se presenta la Tienda Virtual dentro del paradigma de Negocio a Clientes. Así, se propone una Tienda Virtual dedicada a la comercialización de música por Internet, la aplicación llevara el nombre de Music e-Shop y deberá estar conformada por dos partes: el proceso de compra que inicia un cliente y las tareas administrativas de soporte para el funcionamiento del sito. A continuación se describe la operación de las partes de la Tienda.

Proceso de compra

Un visitante al navegar por el sitio Web busca y visualiza los géneros musicales disponibles y sus canciones en el catálogo de la tienda, lee las características técnicas y el precio de dichas canciones. El visitante explora el sitio hasta que elige las canciones que desea y decide incorporarlas al carrito de compras. Mientras las canciones están en el carrito de compras, el visitante puede sacarlos y volverlos a colocar según su criterio, también es posible modificar la cantidad de las canciones seleccionándolas sin necesidad de sacarlas del carrito.

El momento de la compra es cuando el visitante decide dar por concluida la visita a la tienda virtual y llevarse la música de su orden de compra, en este caso transmitiéndolas por Internet a su computadora. En esta fase el comprador debe identificarse, dar sus

datos personales y la dirección de correo electrónico, si no esta inscrito se da de alta en el sistema. El importe de la compra es igual a la suma de los precios de cada canción incluida en el carrito más el IVA total de la compra, dicho IVA es equivalente al 15% de la compra.

El cliente ya sabe a cuanto asciende la compra, aun esta a tiempo para cancelarla si así lo desea. El modo de pago habitual es la tarjeta de crédito por medio de una transacción por Internet utilizando alguno de los servicios disponibles, en este caso se obtiene una respuesta automática que autoriza la operación. Finalmente se genera un recibo/factura que se visualiza en una página que puede ser impresa por el cliente y entonces él puede descargar a su computadora la música de la orden de su compra.

Tareas administrativas

Estas tareas abarcan exclusivamente la administración de las funcionalidades de la aplicación, las cuales no son expuestas a los visitantes de la Tienda. Dichas tareas solo son realizadas por una persona ó un grupo de personas de la tienda (vendedor y/ó personal de la tienda), que actúa como administrador de la aplicación, con los debidos permisos y responsabilidades. En seguida se mencionan las actividades que realiza el administrador de la aplicación:

Colocar a cada canción en distintas géneros jerárquicas para facilitar la navegación y la búsqueda. Se tiene que intentar que la navegación por la tienda sea lo mas ágil posible, por eso la subdivisión es algo a lo que se requiere con frecuencia. El catalogo de canciones es el centro de la aplicación Web y es quizá el elemento que mas cambia. Es habitual la aparición de nuevas canciones y también lo es la descatalogación de canciones que ya no están a la venta. Esto produce las típicas altas y bajas de las canciones, así como las modificaciones ya que las canciones tienen atributos que los describen y que pueden variar como el precio.

Es importante que el administrador tenga la opción de consultar el registro de ventas con el fin de conocer las ganancias generadas y que canciones han tenido más demanda y cuales menos para tomar en cuenta estos datos en el momento de actualizar y gestionar los catálogos de la tienda, además de visualizar su estado económico, Por lo que se requieren reportes convenientes para la toma de decisiones en la gestión de la tienda.

2.2 Modelo de Casos de Uso

El modelo de casos de uso es la base del proceso de desarrollo del sistema. Como se explico en el capítulo anterior un caso de uso en UML es una unidad coherente de la funcionalidad proporcionada por la aplicación que obra recíprocamente con uno o mas actores de la aplicación. Un actor representa una entidad externa que interactúa con el sistema, dicha entidad puede ser un rol desempeñado por los usuarios de la aplicación, una maquina u otro sistema. Para identificar a los actores participantes en el sistema de la tienda virtual, Music e-Shop, a continuación se describen en forma breve:

Cliente.- Es la persona que entra a la Tienda Virtual con la intención de realizar una compra o de simple exploración de canciones y precios. Este usuario puede concretar una compra si así lo desea.

Administrador.- Es la persona que realiza las tareas administrativas de la aplicación para mantener la funcionalidad del sitio.

Entidad bancaria.- Es un sistema que se encarga de realizar las transacciones bancarias necesarias al efectuar una compra en la Tienda Virtual.

En la Figura 2.1 se presenta el diagrama de casos de uso donde se describe la interacción de los actores con el sistema Music e-Shop con el propósito de representar el modelo de las funciones deseadas para Music e-Shop y lo que lo rodea. Un cliente al visitar la tienda ve los géneros musicales disponibles y sus canciones con la opción de elegir las canciones que desea colocándolas en el carrito de compras y modificando su contenido según su criterio, esta función la realiza mediante el caso de uso *Explorar Tienda*, el cual tiene dos extensiones: El caso de uso *Buscar*, cuando el visitante realiza una búsqueda de música. Y el caso de uso *Comprar* para efectuar la compra de las canciones que el visitante introdujo en el carrito para ello incluye a los casos de uso *Autenticar Usuario* y *Realizar Pago*. Con el primero se identifica un cliente y en caso de que no este registrado en el sistema se da de alta mediante el caso de uso *Registrar Cliente*, el administrador también se identifica con el caso de uso *Autenticar Usuario*. Con el segundo, *Realizar Pago*, el cliente puede llevar a cabo el pago de su compra a través de la *Entidad Bancaria*, la cual realizara las operaciones necesarias mediante el caso de uso *Ejecutar Transacción Bancaria*. Una vez que se concluye el pago se da por valida la compra y se registra en el sistema mediante el caso de uso *Ejecutar Venta* y se muestra una página con la factura de compra que el cliente puede imprimir mediante

el caso de uso *Imprimir Factura*. Al concretar la compra el usuario puede adquirir sus canciones mediante el caso de uso *Descargar Música*. Finalmente para el buen funcionamiento de la Tienda el administrador puede actualizar el catálogo de canciones mediante el caso de uso *Gestionar Catálogo* y ver los registros de ventas de música, del catálogo y los clientes mediante el caso de uso *Consultar Reportes*.

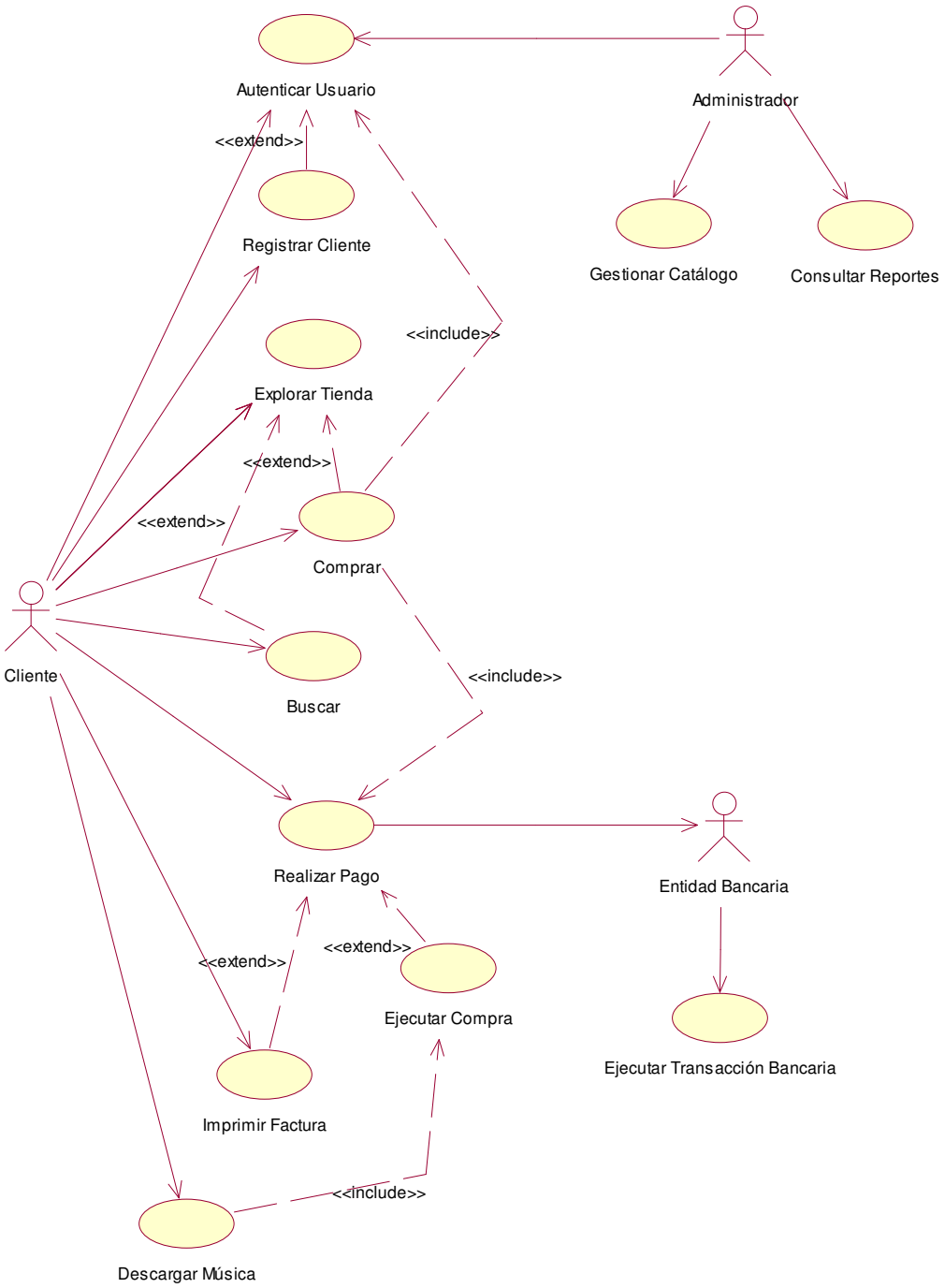

Figura 2.1: Diagrama de Casos de Uso para Music e-Shop.

2.3 Comportamiento de los Casos de Uso Mediante Diagramas de Actividades

Los diagramas de actividades pueden ser creados en esta etapa de desarrollo del sistema Music e-Shop. Estos diagramas representan el dinamismo del sistema y son gráficos de flujos que se utilizan para mostrar el flujo de trabajo de un sistema; es decir, muestran el flujo de control de una actividad a otra en el sistema, qué actividades pueden ser realizadas en paralelo y cualquier otra ruta alternativa sobre el flujo. En esta fase de análisis, los diagramas de actividades son creados para representar el flujo sobre los casos de uso o bien para representar el flujo dentro de un caso de uso particular. En fases posteriores del desarrollo sirven para mostrar el flujo de trabajo para una determinada operación.

Un diagrama de actividades contiene actividades, transiciones entre las actividades, puntos de decisión y barras de sincronización. A continuación se describen algunos de los diagramas de actividades que representan el comportamiento de los casos de uso de Music e-Shop.

2.3.1 Actividades para el Caso de Uso “Explorar Tienda”

En el diagrama de actividades de la Figura 2.2 se describe el comportamiento del caso de uso *Explorar Tienda* con el que interactúa un Cliente para navegar en Music e-Shop. Cuando un usuario entra a Music e-Shop a explorar la tienda con ánimo de comprar, puede inicialmente intentar una búsqueda, seleccionar un género musical del catálogo o ver el carrito de compras. Con las dos primeras actividades el cliente puede ver la música de la tienda; así, si el Cliente selecciona un género musical podrá visualizar las canciones que pertenecen a dicho género, incluyendo sus características y precio. Luego puede seleccionar una o más canciones de la lista, si así lo desea, para agregarlas al carrito de compras. Por el contrario, si no desea adicionar canción alguna al carrito puede seleccionar otro género musical del catálogo y continuar con este proceso ó dar por concluida la exploración. De otro modo, si el Cliente intenta una búsqueda y esta concluye, entonces se despliega la lista de los resultados, de donde puede seleccionar las canciones que desee agregar al carrito. En el caso de que el Cliente haya elegido agregar una o más canciones al carrito, ya sea mediante la elección de un género o una búsqueda, visualizara el contenido del carro; es decir, la

lista de canciones, sus respectivos precios y el total a pagar de las canciones que ha incorporado durante la visita actual a la tienda. Además, el cliente modifica el contenido del carro eliminando una o más canciones según su criterio, para ello elige la o las canciones a eliminar y actualiza el carrito para ver las canciones restantes y su correspondiente monto total. Desde el carrito de compras el cliente decide si sigue explorando canciones de los géneros musicales, pasa a ordenar para hacer efectiva su compra o simplemente salir de la tienda. Cabe señalar que las actividades Buscar y Comprar solo son representaciones de los casos de uso que llevan su nombre. En las siguientes secciones se explicaran el comportamiento de estos casos de uso mediante sus correspondientes diagramas de actividades.

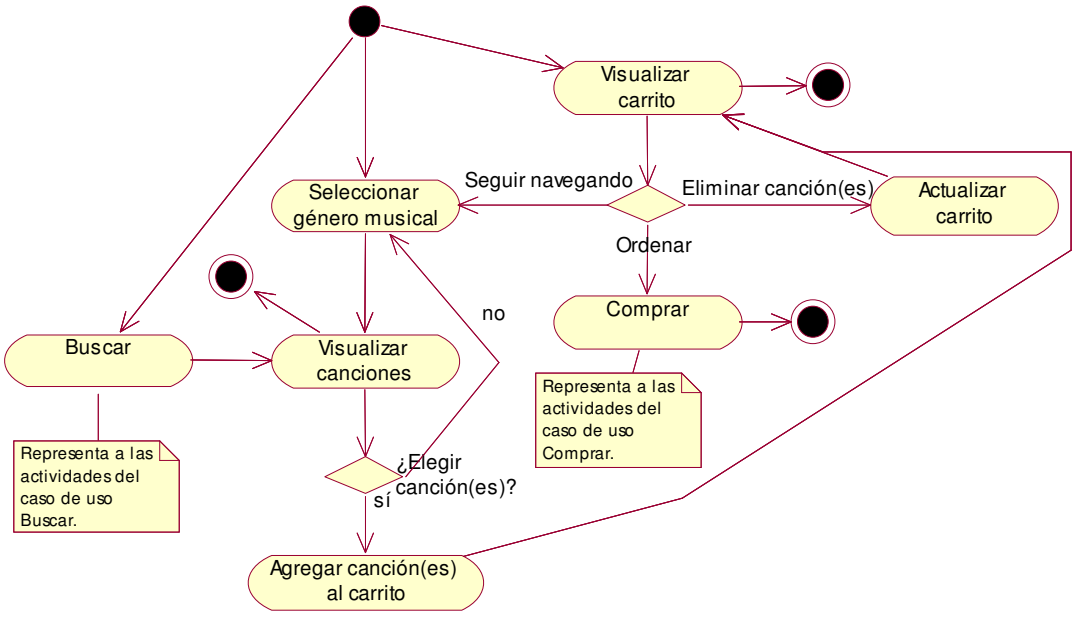

Figura 2.2: Diagrama de Actividades para el Caso de Uso Explorar Tienda.

2.3.2 Actividades para el Caso de Uso "Buscar"

En el diagrama de actividades de la Figura 2.3 se describe el comportamiento del caso de uso *Buscar* con el que interactúa un Cliente para encontrar canciones en Music e-Shop. En la sección anterior se menciono que un Cliente puede, inicialmente, intentar una búsqueda al navegar en Music e-Shop, entre otras actividades. En este caso, el cliente puede buscar música, determinada por el titulo de una canción o por el nombre de un artista. Así, el Cliente selecciona el parámetro de búsqueda, canción o artista, e

introduce el nombre correspondiente a su selección. Una vez seleccionado el parámetro e introducido su nombre, el Cliente elige iniciar búsqueda. Al concluir la búsqueda se muestran sus resultados, ya sea una lista de las canciones o un mensaje que indica que no se encontró canción alguna del parámetro introducido. En el primer caso, el Cliente visualizara las canciones asociadas al parámetro introducido. Si le interesa una o más de ellas, podrá agregarlas al carrito de compras siguiendo el proceso del diagrama anterior (Secc. 2.3.1, Figura 2.2). El Cliente puede intentar varias búsquedas mientras explora Music e-Shop y concluir su visita en cualquier momento.

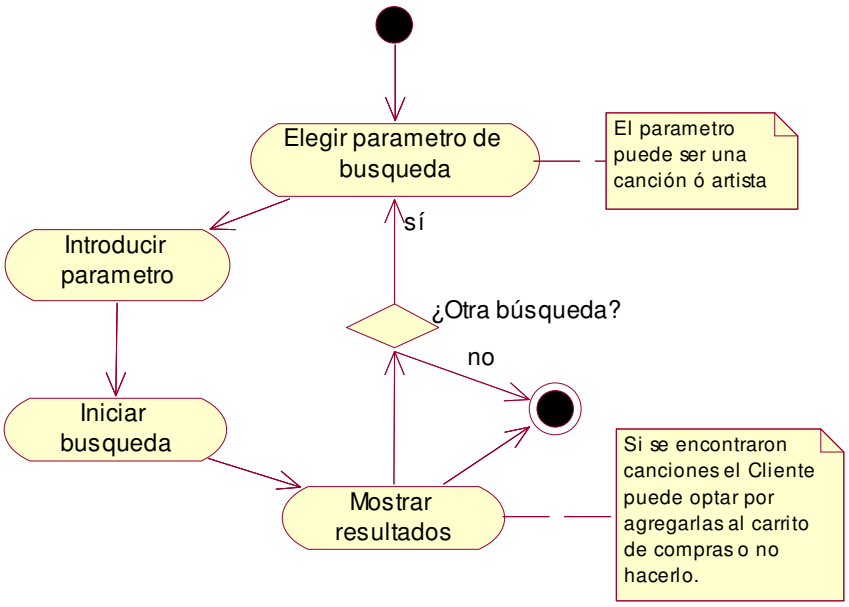

Figura 2.3: Diagrama de Actividades para el Caso de Uso Buscar.

2.3.3 Actividades para el Caso de Uso “Comprar”

En este diagrama de actividades, Figura 2.4, se describe el comportamiento del caso de uso *Comprar*, con el que interactúa un Cliente en Music e-Shop, para efectuar la compra de las canciones que introdujo en el carrito. El prerrequisito para que el cliente pueda efectuar una compra es que exista por lo menos una canción en el carrito de compras y que el Cliente pase a ordenar el contenido del carrito. Una vez cumplido el prerrequisito, el comprador debe identificarse ya sea si es un Cliente existente o un nuevo Cliente. Si es un Cliente existente, se identifica introduciendo su e-mail y contraseña; si su autenticación es incorrecta tendrá que intentar identificarse

nuevamente. De otro modo, el comprador, como nuevo Cliente, se registra introduciendo sus datos personales, e-mail y contraseña. La identificación de un Cliente se efectúa mediante el flujo de actividades del caso de uso *Autenticar Usuario* para un Cliente existente y del caso de uso *Registrar Cliente* para un nuevo Cliente. Los cuales se muestran en forma representativa, en este diagrama. Al efectuarse la autenticación o el registro exitosamente se despliega la información de los datos de facturación del cliente, los detalles de su orden de compra y un formulario de pago. El Cliente puede abortar su compra o llenar el formulario al introducir los datos de la tarjeta de crédito con la que efectuara el pago de su orden. Finalmente el Cliente elige pagar para solventar el monto total de su orden a través de la Entidad Bancaria. Esto se lleva a cabo mediante el flujo de actividades del caso de uso *Realizar Pago*, representado en la Figura 2.4 por la actividad que lleva su nombre; su comportamiento se explica en la siguiente sección.

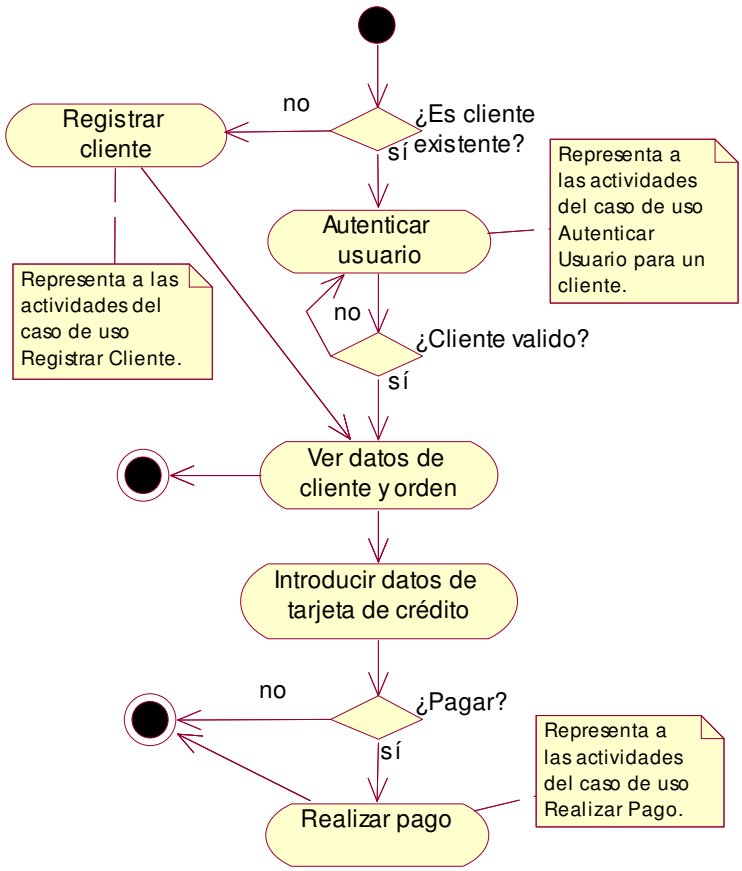

Figura 2.4. Diagrama de Actividades para el Caso de Uso Comprar.

2.3.4 Actividades para el Caso de Uso "Realizar Pago"

En este diagrama de actividades, Figura 2.5, se describe el comportamiento del caso de uso *Realizar Pago* que realiza un Cliente en Music e-Shop para llevar a cabo el pago de su compra a través de la Entidad Bancaria. El Cliente envía la información de pago, datos de su tarjeta de crédito, a la Entidad Bancaria. La cual realizara las operaciones necesarias mediante el caso de uso *Ejecutar Transacción Bancaria*. La Entidad Bancaria es la que determina si el resultado del pago fue exitoso o no; en caso de que no sea exitoso enviara un mensaje de error que podrá observar el Cliente. De lo contrario el Cliente visualiza la confirmación del pago como exitoso y su factura. Al mismo tiempo que se da por valida la compra y se registra en el sistema mediante el caso de uso *Ejecutar Venta*; representado en la Figura 5 con la actividad que lleva su nombre. Al hacer efectiva la compra el usuario puede descargar la música de su orden de compra e imprimir su factura como una opción adicional. Estas dos últimas actividades son la representación de los flujos correspondientes a los casos de uso *Imprimir Factura* y *Descargar Música*; con las cuales concluye el proceso de compra.

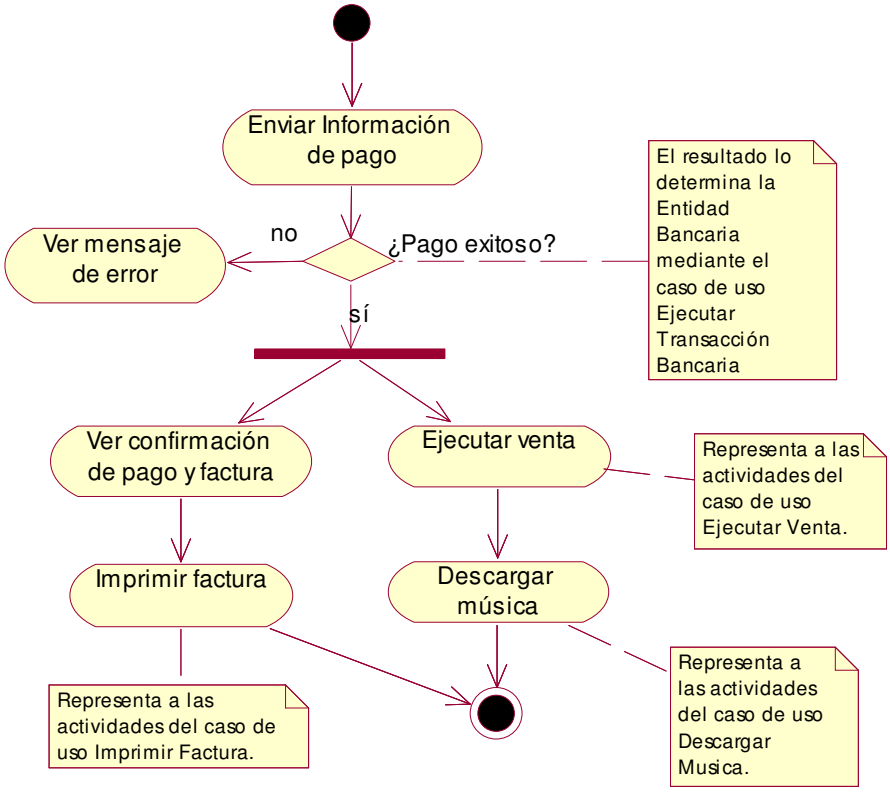

Figura 2.5. Diagrama de Actividades para el Caso de Uso Realizar Pago.

2.3.4 Actividades para el Caso de Uso "Gestionar Catálogo"

En este diagrama de actividades, Figura 2.6, se describe el comportamiento del caso de uso Gestionar Catálogo que realiza el Administrador para agregar y organizar la música en el catálogo de Music e-Shop. El Administrador, desde el modulo Gestión Catálogo, puede agregar una canción nueva o bien, un género nuevo al catálogo. A partir del listado de todos los géneros existentes en el catalogo de Music e-Shop, para cada genero, se encuentran opciones para solicitar la eliminación del género, la edición del mismo o la visualización de las canciones que contiene. Así, se listan todas las canciones que contiene el género correspondiente. Para cada canción del listado existen opciones para solicitar la eliminación de la canción, editarla o moverla a un género. El Administrador introduce los datos requeridos para llevar a cabo estas tareas para el registro de canción nueva, modificación de una canción existente o bien, movimiento de canción. El Administrador, para los géneros, cuenta con las operaciones para dar de alta a un nuevo género en el catálogo, modificar el nombre de uno existente o bien, eliminarlo.

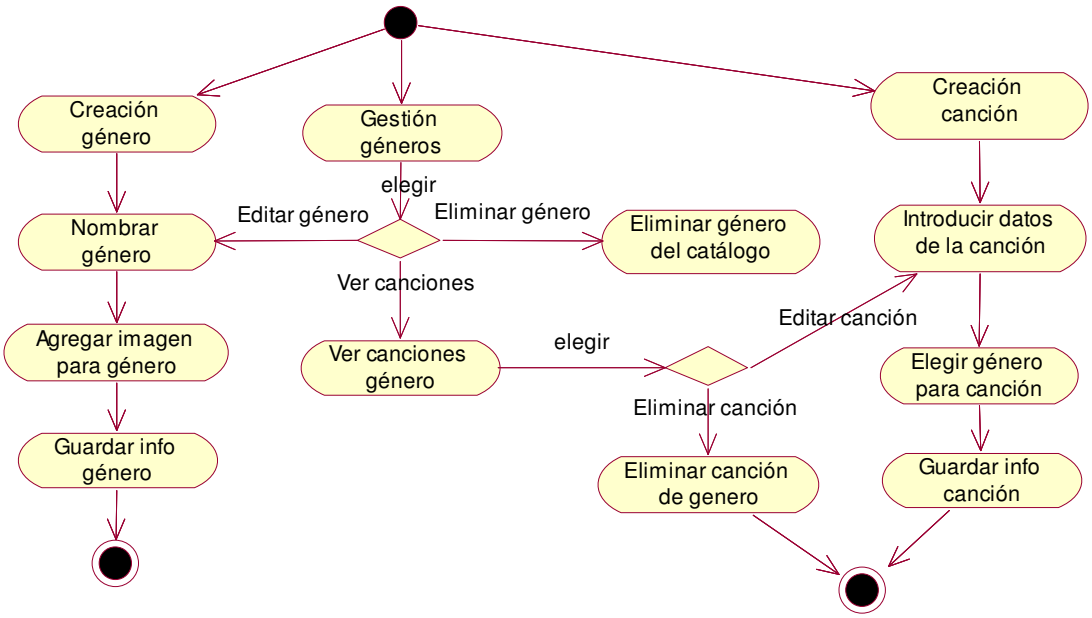

Figura 2.6: Diagrama de Actividades para el Caso de Gestionar Catálogo.

Capítulo 3

Diseño de Music e-Shop

En este capítulo se presenta el diseño del sistema para Music e-Shop. En el capítulo anterior se presento el primer diagrama de la metodología UWE correspondiente al análisis del sistema, la especificación de requerimientos. Ahora, como parte del diseño de la aplicación, se presenta el resto de los diagramas de UWE para completar el modelado de Music e-Shop. Así, en el primer apartado se presenta el modelo lógico conceptual de Music e-Shop, el cual describe una primera abstracción de los elementos de la aplicación Web mediante un diagrama de clases simple de UML. En el segundo apartado se muestra el modelo de navegación, el cual consiste en llevar a cabo un diagrama de clases estereotipado que muestre la manera en que los usuarios pueden navegar dentro de la aplicación. Continuando en el tercer apartado con la interacción temporal la cual describe las secuencias de Music e-Shop. Y finalmente en el cuarto se presenta el escenario Web de la aplicación mediante diagramas de transición de estados, para describir las acciones necesarias de funcionalidades provistas por Music e-Shop.

3.1 Modelo Lógico Conceptual

En el modelo lógico conceptual, como se menciono anteriormente, se exponen asociaciones entre elementos de la aplicación a desarrollar, haciendo énfasis en el flujo lógico entre elementos, en su semántica y evitando hacer referencia en la medida de lo posible de cuestiones relacionadas con la navegación y de los aspectos de interacción.

En la Figura 3.1 se muestra un diagrama de clases simple de UML en donde se presentan los principales elementos de Music e-Shop y la asociación entre ellos, es decir, un modelo lógico-conceptual que representa de forma general la estructura de Music e-Shop. En el se observan clases, multiplicidad y asociaciones de agregación y composición. La semántica del diagrama presenta estas clases como elementos involucrados en las actividades típicas que el usuario realizará con la aplicación Web, y las asociaciones entre ellos como relaciones de cooperación que, en algún momento, ha de llevarse a cabo para realizar una tarea. En el diagrama se puede observar que el sistema Music e-Shop esta compuesto por la Tienda y su panel de Administración. A la

tienda le corresponde catálogo de música determinado en géneros musicales que contienen un número determinado de canciones. También cuenta con un buscador de música para encontrar determinado tipo de canciones. Una o más canciones pueden ser agregadas a un carrito de compras. A cada orden de un Carrito se asocia un Formulario de Pago a través del cual se envía la información de pago hacia el Gestor de Transacciones. El resultado de las transacciones, la generación de los registros de compra y su factura correspondiente, se incluyen. También se incluye el Autenticador de Usuarios; este módulo es el encargado de realizar la autenticación de los diferentes tipos de usuario para permitir o no el acceso a sus correspondientes partes del sistema mediante su Sesión de Usuario. Así también, la tienda tiene varios usuarios asociados, todos los cuales son administrados por el padrón de usuarios.

En lo que se refiere a la parte de Administración se muestran las diferentes asociaciones con los módulos disponibles. De esta forma incluye un gestor del catálogo de la tienda, y un módulo para la visualización de informes de las ventas, usuarios y el catálogo. En el siguiente capítulo se presenta el modelo de navegación tomando como referencia este modelo lógico-conceptual.

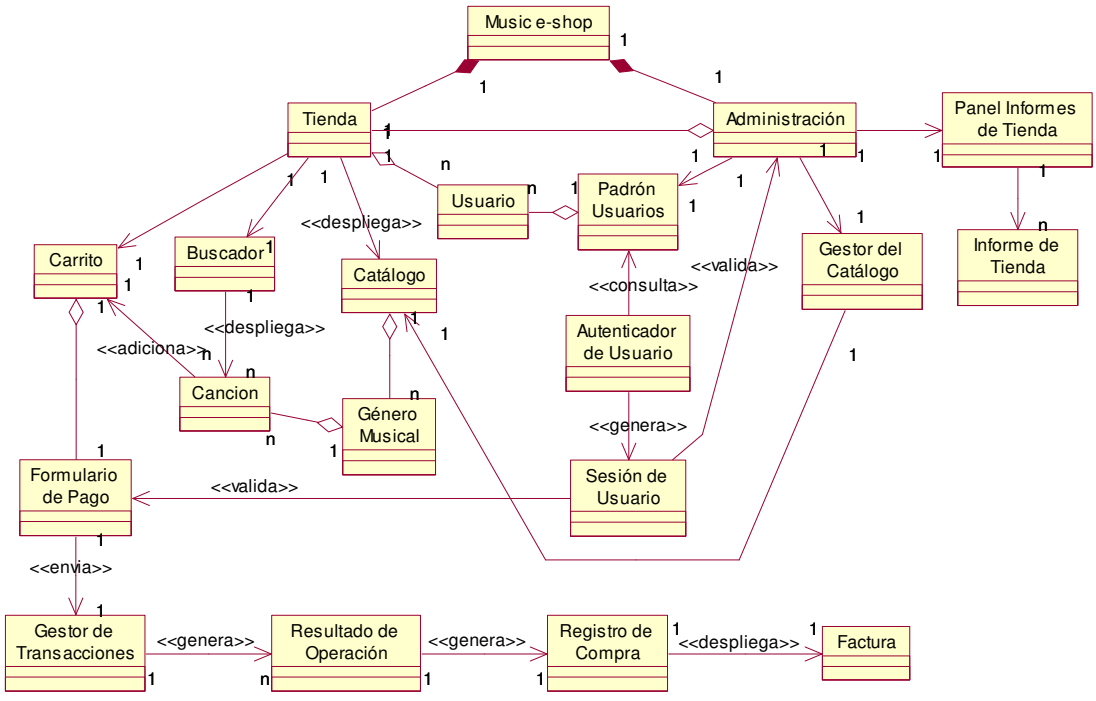

Figura 3.1 Modelo Lógico-Conceptual de Music e-Shop.

3.2 Modelo de Navegación

El modelo de navegación comprende la especificación de qué objetos pueden ser visitados mediante la navegación en la aplicación Web y las asociaciones entre ellos. Éste modelo se representa por los diagramas de clases estereotipadas. En las dos siguientes secciones se presenta el modelo de navegación para Music e-Shop. En la primera se muestra la Navegación en Music e-Shop enfocada al proceso de la compra de un Cliente e incluye la autenticación de un usuario administrador. Y en la segunda sección se expone la navegación en la administración de Music e-Shop dirigida a las tareas de gestión de la tienda virtual. Antes de continuar se citan algunos estereotipos utilizados según [RossOcaCCV06]:

- El estereotipo <<navigation class>> que representa una clase conceptual asociada con una página Web.
- El estereotipo <<form>> aplicable a una clase y que representa un formulario.
- El estereotipo <<index>> también aplicable a una clase y que representa un índice de elementos dentro de una página Web
- El estereotipo <<link>> aplicable a una asociación simple unidireccional en un diagrama de clases y que representa un hipervínculo que liga una página Web con otra.
- El estereotipo <<submit>> aplicable a una asociación simple unidireccional en un diagrama de clases y que representa una relación direccional entre un <<form>> y una <<navigation class>>
- El estereotipo <<redirect>> aplicable a una asociación simple unidireccional en un diagrama de clases y que representa el paso directo de una <<navigation class>> a otra sin que el usuario lo solicite.

3.2.1 Navegación en Music e-Shop

En la Figura 3.2 se muestra el diagrama de clases estereotipado correspondiente al modelo de navegación de Music e-Shop. Dicho modelo tiene una <<navigation class>> principal que es *Home Music e-Shop*. A partir de ella, todas las demás que se describen en el modelo son navegables en algún momento en la realización de una tarea. A

Home Music e-Shop no le precede ninguna <<navigation class>>, puesto que a esta se accede mediante la introducción directa de la dirección Web en la barra de URL del navegador Web de una computadora. El mismo caso aplica para la <<navigation class>> *Login Admon* que es la interfase para la autenticación de usuarios Administradores para acceder a la parte de Administración de Music e-Shop.

La navegación en Music e-Shop a través de su catálogo de música, se provee con las <<navigation class>> *Home Music e-Shop*, *Género* y los <<index>> *Catálogo* y *Canción*. En *Home Music e-Shop* se carga el catálogo de música, el cual se organiza y presenta al Cliente mediante el <<index>> *Catálogo* y la <<navigation class>> *Género*. *Catálogo* es un listado de los géneros musicales en los que se clasifican las canciones. Mientras que en *Género se cargan las canciones, especificaciones y precio de dichas canciones*; mediante el index *Canción*. *Canción* esta compuesto por el listado de las canciones que contiene el género. Desde la <<navigation class>> *Genero* se puede navegar hacia la <<navigation class>> *Carrito* mediante la adición de una o mas canciones o bien, navegar hacia *Home Music e-Shop* para explorar alguna otra sección del catálogo de la tienda.

La gestión del carrito de compras y las ordenes de compra se realiza con la <<navigation class>> *Carrito*, el <<index>> *Listado Canciones* y los <<form>> Elegir Articulo y *Actualizar carro*. En *Carrito* se definen las operaciones de adición y eliminación de una o más canciones en una orden. Una o mas canciones, previamente seleccionadas por el usuario, se añaden simultáneamente, mediante una solicitud enviada desde la <<navigation class>> *Genero* hacia la <<navigation class>> *Carrito*. En *Carrito* además aparece asociado mediante agregación el <<index>> *Listado Canciones*, en el cual se enumeran todas las canciones adicionadas al momento en la orden. A cada elemento del listado de canciones adicionadas en la orden del carrito se asocia también por composición un <<form>> Elegir Articulo. El propósito de este formulario junto con el <<form>> *Actualizar Carro* es el de servir de interfase para que el Cliente pueda marcar una o mas canciones previamente adicionadas para que sean eliminadas de la orden. Los datos manipulados a través de estos <<form>> se envían a la misma <<navigation class>> *Carrito* para que se hagan efectivos los cambios realizados por el Cliente. A partir de *Carrito*, el Cliente puede elegir entre continuar de compras en la Tienda, o bien, inicializar el proceso de pago e ir a *Login Cliente*. Cabe

mencionar que se asocia una orden temporal sobre la cual se realiza la gestión del carrito para cada Cliente.

Cuando un Cliente desea iniciar el proceso de pago, pueden darse dos situaciones: que sea la primera vez que accede a Music e-Shop a comprar o bien que ya se encuentre registrado. La funcionalidad del registro del Cliente nuevo se provee con la <<navigation class>> *Registro Cliente* y el <<form>> *Datos Cliente*. *Registro Cliente* es navegable a partir de la <<navigation class>> *Login Cliente*, para inicia el proceso de pago, y en ella se encuentra definida la operación de registro de Cliente nuevo. A *Registro Cliente* se asocia por composición el <<form>> *Datos Cliente* que es el formulario a través del cual el Cliente introduce sus datos generales, e-mail y contraseña. *Registro Cliente* recibe esta información, la valida y realiza el registro del Cliente. Después del registro, el proceso de pago continua re direccionando la navegación hacia la <<navigation class>> *Pago de Música*. Cabe mencionar que *Registro Cliente* también es navegable a partir de la <<navigation class>> *Pago de Música* debido a que en *Registro Cliente* se encuentran definidas las operaciones de actualización de datos del Cliente.

En el caso de que el Cliente ya se encuentre registrado, entonces, para iniciar el proceso de pago, debe introducir sus datos de registro, que son su correo electrónico y contraseña. Esta funcionalidad se provee mediante el <<form>> *Datos Registro Cliente* asociado a la <<navigation class>> *Login Cliente*. Este par de datos se envía hacia la <<navigation class>> *Autenticación Usuario* en donde se realiza la verificación de los datos del registro del Cliente. Si los datos de registro son correctos entonces el proceso se reanuda en la <<navigation class>> *Pago de Música*. La funcionalidad de *Pago de Música* es la de presentar al Cliente un resumen de la orden sobre el cual se va proceder a realizar la transacción y la información de registro del Cliente. Además, el proceso de pago se completa mediante la <<navigation class>> *Pago de Música* a la cual esta asociada por composición el <<form>> *Info Pago Música*. Desde *Pago de Música* el Cliente tiene la posibilidad de requerir la modificación de los datos de facturación, funcionalidad que se encuentra provista por la <<navigation class>> *Registro Cliente*. La última fase para la requisición del pago del pedido se realiza mediante el <<form>> *Info Pago*. A través de este, el Cliente introduce la información de la tarjeta bancaria sobre la cual se debe realizar el cargo de su orden de compra. Esta información es la que se envía hacia la entidad bancaria para la ejecución de la

operación mediante la gestión de la transacción por parte del subsistema Gestor de Transacciones representado en el modelo. De acuerdo al resultado de la operación, éxito o fracaso, la respuesta obtenida del Gestor de Transacciones se presenta en la <<navigation class>> *Confirm Pago Música* o en la <<navigation class>> *Error Pago Música*, acorde a dicho resultado.

Por último, la autenticación de los dos diferentes tipos de usuarios de Music e-Shop se realiza mediante las <<navigation class>> *Autenticación Usuario*, *Login Cliente*, *Login Administrador* y los <<form>> *Datos Registro Cliente* y *Datos Registro Administrador*. Las interfases para la autenticación de cada tipo de usuario se proveen mediante la <<navigation class>> correspondiente a cada tipo de usuario con su respectivo <<form>> asociado por composición, como se observa en el modelo. Los <<form>> se utilizan para que por medio de ellos los usuarios introduzcan sus datos de registro. El procedimiento de autenticación para cualquier tipo de usuario se realiza en la <<navigation class>> *Autenticación Usuario*. Esta <<navigation class>> es de suma importancia para Music e-Shop ya que en ella encuentran definidas las operaciones para verificar, por medio de la comprobación de los datos de registro, si un usuario ya sea Cliente o Administrador de Music e-Shop se encuentra inscrito en el padrón de usuarios del sistema. *Autenticación Usuario* verifica que los datos de registro que son enviados desde las interfases correspondientes al logeo de Clientes y Administradores, correspondan con los datos que se encuentran resguardados en la base de datos de Music e-Shop. Si la verificación es correcta, entonces *Autenticación Usuario* reanuda la navegación hacia la <<navigation class>> que corresponda al tipo de usuario que se autenticó, de lo contrario la navegación se reanuda solicitando de nueva cuenta los datos de registro del usuario.

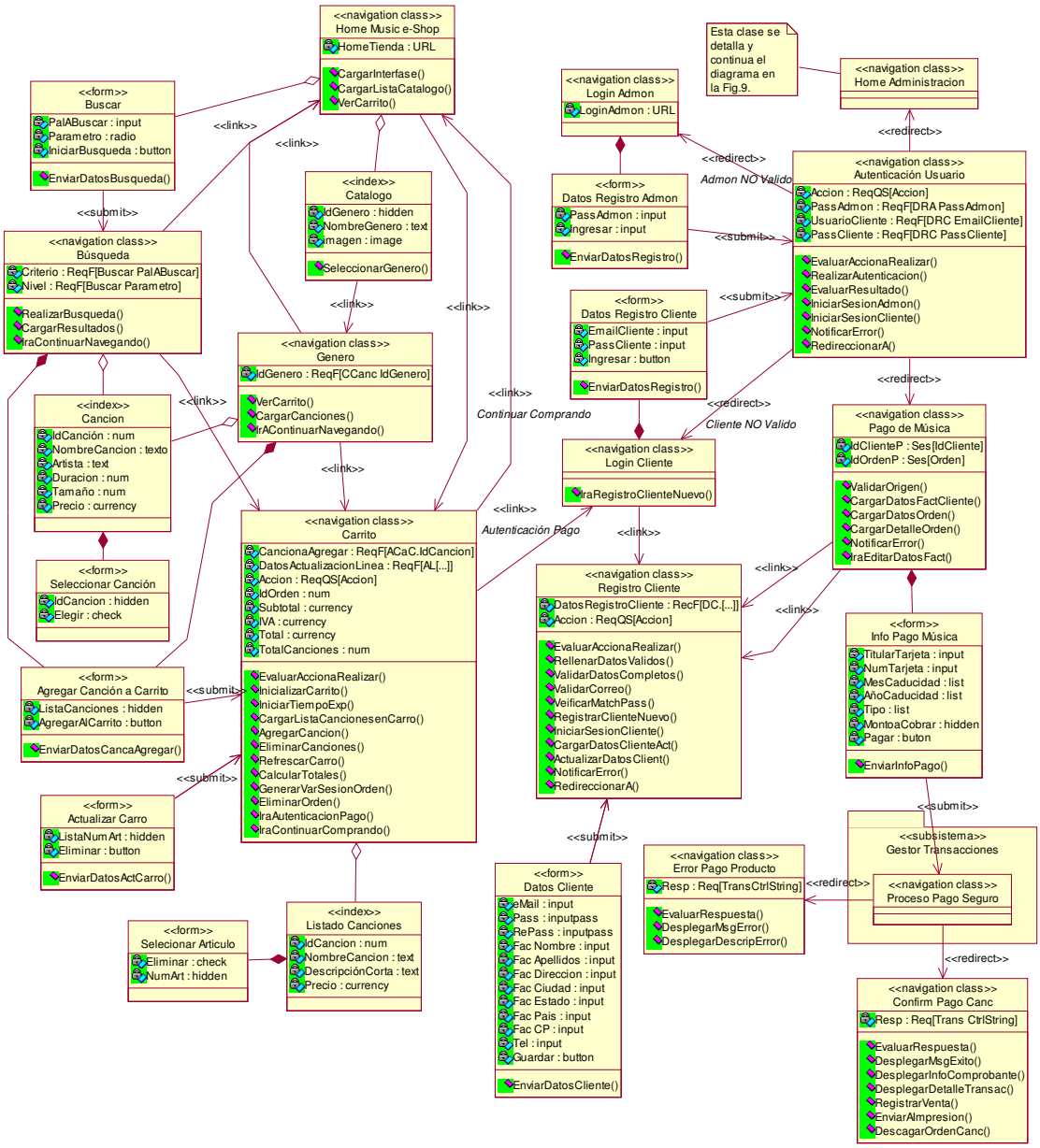

Figura 3.2: Modelo de Navegación de Music e-Shop, parte 1.

3.2.2 Navegación en la Administración de Music e-Shop

En la Figura 3.3 se muestra la continuación del modelo de navegación de Music e-Shop que corresponde a la navegación en la administración de Music e-Shop por un usuario Administrador. Una vez que un usuario Administrador se ha validado correctamente en Music e-Shop, accede a la <<navigation class>> principal de la Gestión de la Tienda

que es *Home Administración*, a partir de la cual se puede navegar hacia los módulos Informes y Gestión Catalogo. Las principales funcionalidades de la administración son: el manejo del catálogo de música y la visualización de informes de Music e-Shop. Estas funcionalidades se proveen mediante clases estereotipadas y asociaciones que a continuación se describen.

La gestión del catálogo de canciones es la funcionalidad esencial de la administración. Esta opción permite al Administrador organizar la música; para ello, participan las <<navigation class>> *Gestión Catálogo*, *Registro Canción* y *Registro Género*, los <<index>> *Catálogo* y *Género* y los <<form>> *Datos Canción* y *Datos Género*. El Administrador, a partir de *Gestión Catálogo,* puede navegar hacia la <<navigation class>> *Registro Canción* para agregar una canción nueva o bien, navegar hacia la <<navigation class>> *Registro Género* para agregar un género nuevo al catálogo. *Gestión Catálogo* tiene asociado el <<index>> *Catálogo*, el cual es un listado de todos los géneros existentes en el catalogo de Music e-Shop. En cada género de este listado se encuentran opciones para solicitar la eliminación del género, la edición del mismo o la visualización de las canciones que contiene a través del <<index>> *Género* en donde se listan todas las canciones que contiene el género correspondiente. Para cada canción del listado existen opciones para solicitar la eliminación de la canción, editarla o moverla de género. En la <<navigation class>> *Registro Canción* se encuentran definidas las operaciones para registrar una canción nueva, modificarla, eliminarla o moverla a otra género. El Administrador introduce los datos requeridos para llevar a cabo estas tareas a través del <<form>> *Datos Canción* para el registro de canción nueva y para la modificación de una canción existente. En la <<navigation class>> *Registro Género* se encuentran definidas las operaciones para dar de alta a un nuevo género en el catálogo, modificar el nombre de uno existente o bien, eliminarlo.

Para dar la posibilidad al Administrador de cuantificar la actividad de la tienda virtual, la Administración de Music e-Shop incluye un gestor de Informes. Esta funcionalidad se provee mediante las <<navigation class>> *Informes*, *Informe Tienda* y el <<index>> *Opciones de Informe*. En *Informes* se presentan los tipos de informes que se encuentran disponibles. Cada tipo contiene un grupo de opciones de visualización, el cual se presenta mediante el <<index>> *Opciones de informe*. Cuando el Administrador ha seleccionado un informe en específico este se presenta en la <<navigation class>> *Informe Tienda*.

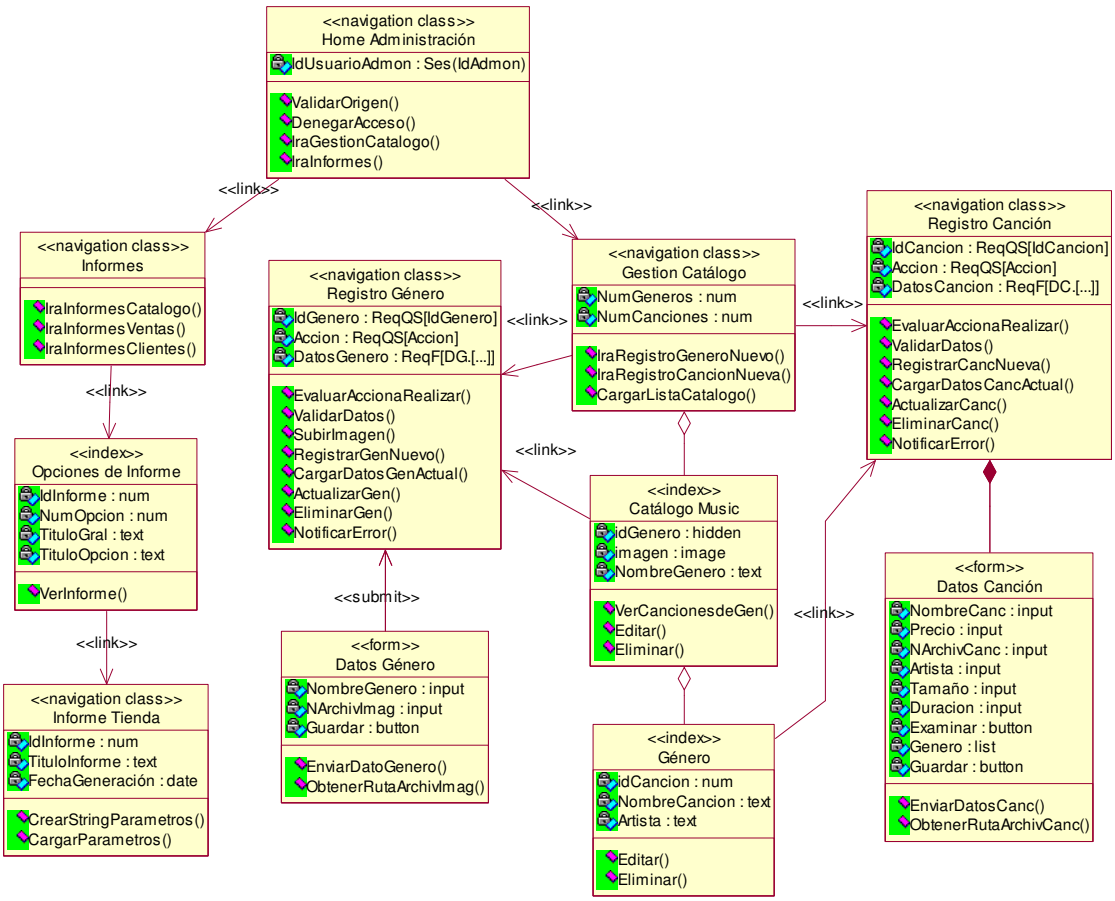

Figura 3.3 Modelo de Navegación de Music e-Shop, parte 2.

3.3 Interacción Temporal

En esta sección se presentan las interacciones en orden temporal entre las diferentes clases estereotipadas que aparecen en el modelo de navegación de la aplicación. En las siguientes sub secciones se listan los diagramas de secuencia correspondientes al proceso de compra y a la administración de Music e-Shop.

3.3.1 Secuencia Adición de Canciones al Carrito

La secuencia de la Figura 3.4 describe el procedimiento para la adición de un canción al carrito de compras. Esta secuencia comienza con el acceso por parte del Cliente a Music e-Shop mediante la introducción de la dirección URL de *Home Music e-Shop*, la <<navigation class>> inicial. Home Music e-Shop carga la interface de Music e-Shop

con la operación CargarInterface() y carga el Listado de Géneros Musicales con la operación CargarListaCatalogo(). A continuación el Cliente selecciona uno de los géneros del catálogo y al hacer esto, se despliega el listado de las canciones que contiene dicho género. El listado incluye las especificaciones y precio de las canciones que pertenecen al género musical seleccionado. El Cliente ahora selecciona agregar una o más canciones al carrito, esto mediante la operación EnviarDatosCancaAgregar() definida en el <<form>> *Agregar Canción a Carrito*. Cuando esto se realiza los datos del canción se envían a la <navigation class>> *Carrito*. Cuando es el primera vez que un Cliente en especifico esta añadiendo artículos al carrito se deben realizar las operaciones de InicializarCarrito(), GenerarIdOrden(), GenerarVarSesIdOrden() e IniciarTiempoExp(). Con estas operaciones se inicializa el carrito de compras, se genera un identificador de la orden, se genera una variable de sesión con el identificador de la orden, y se inicializa el tiempo de caducidad de las variables generadas. El identificador de la orden y su depósito en una variable de sesión se utiliza como mecanismo de control para asociar a cada orden con el Cliente correspondiente en un periodo de tiempo específico. Esta serie de operaciones únicamente se realizan cuando un Cliente agrega su primera canción al carrito. Indistintamente el procedimiento de adición de canción al carrito continúa con la operación AgregarCancion(), CargarListaCancionesenCarro() y CalcularTotales(). Con estas operaciones se realiza la adición física de música al carrito, se actualiza la lista de las canciones que ya se encuentran en el carrito y se calculan los nuevos totales. Por último, el Cliente puede elegir entre continuar comprando regresando al Home Music e-Shop, o bien elegir iniciar el proceso de pago de la orden con la Autenticación del Cliente.

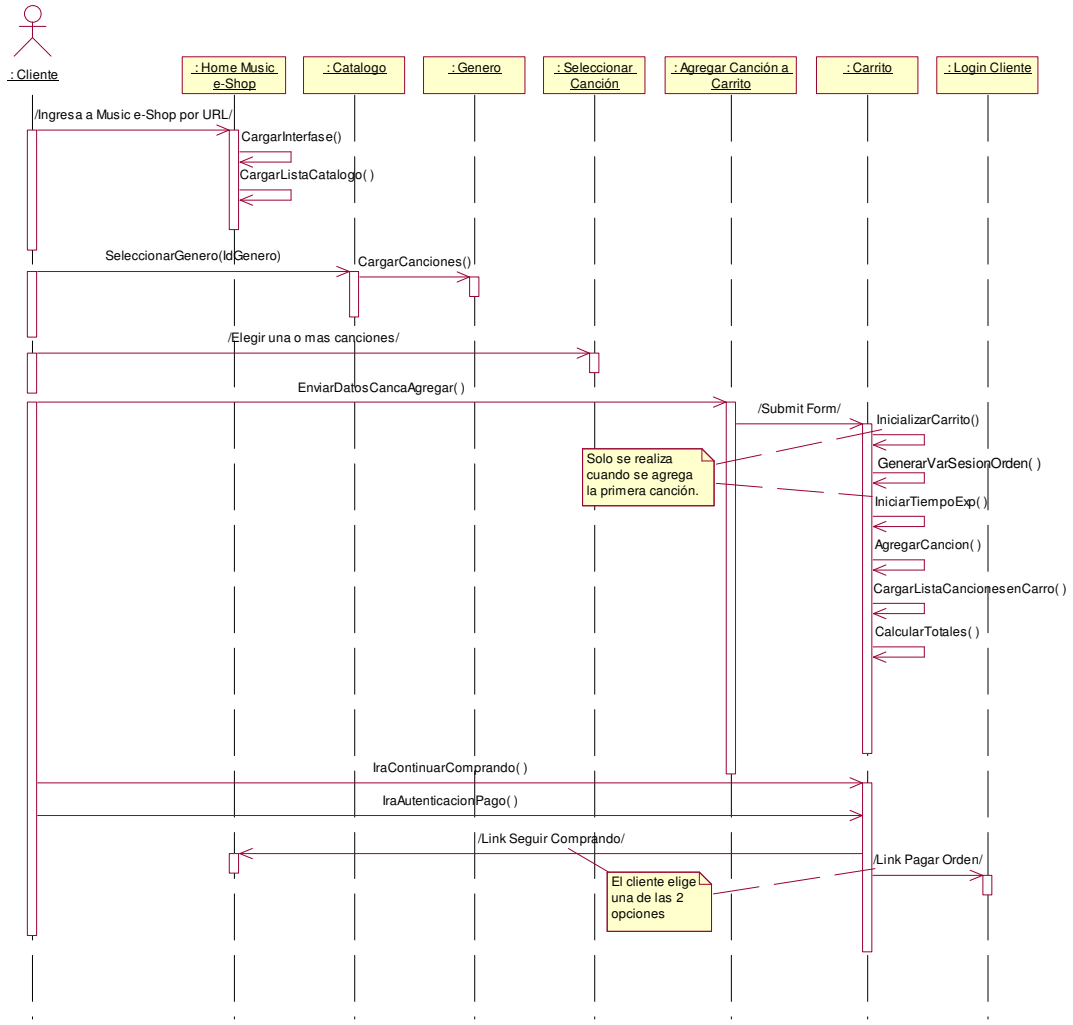

Figura 3.4: Diagrama de Secuencia para la Adición de Canciones al Carrito

3.3.2 Secuencia Búsqueda de Canciones

La secuencia para la búsqueda de una canción en Music e-Shop se muestra en la Figura 3.5. La secuencia comienza cuando el Cliente ingresa a Music e-Shop. El <<form>> *Buscar* se asocia a Home Music e-Shop. El Cliente elige uno de los dos parámetros de búsqueda, canción ó artista, e introduce el nombre del parámetro elegido. Y solicita dicha búsqueda mediante la operación del <<form>> EnviarDatosBusqueda(). El parámetro de la búsqueda es enviado a la <<navigation class>> *Búsqueda* en donde se realiza la búsqueda en las fuentes de datos mediante la operación RealizarBusqueda() para posteriormente cargar los resultados en el

<<index>> *Canción*. Cabe mencionar que el cliente puede seleccionar una o más canciones para agregarlas al carrito de compras de la lista de resultados.

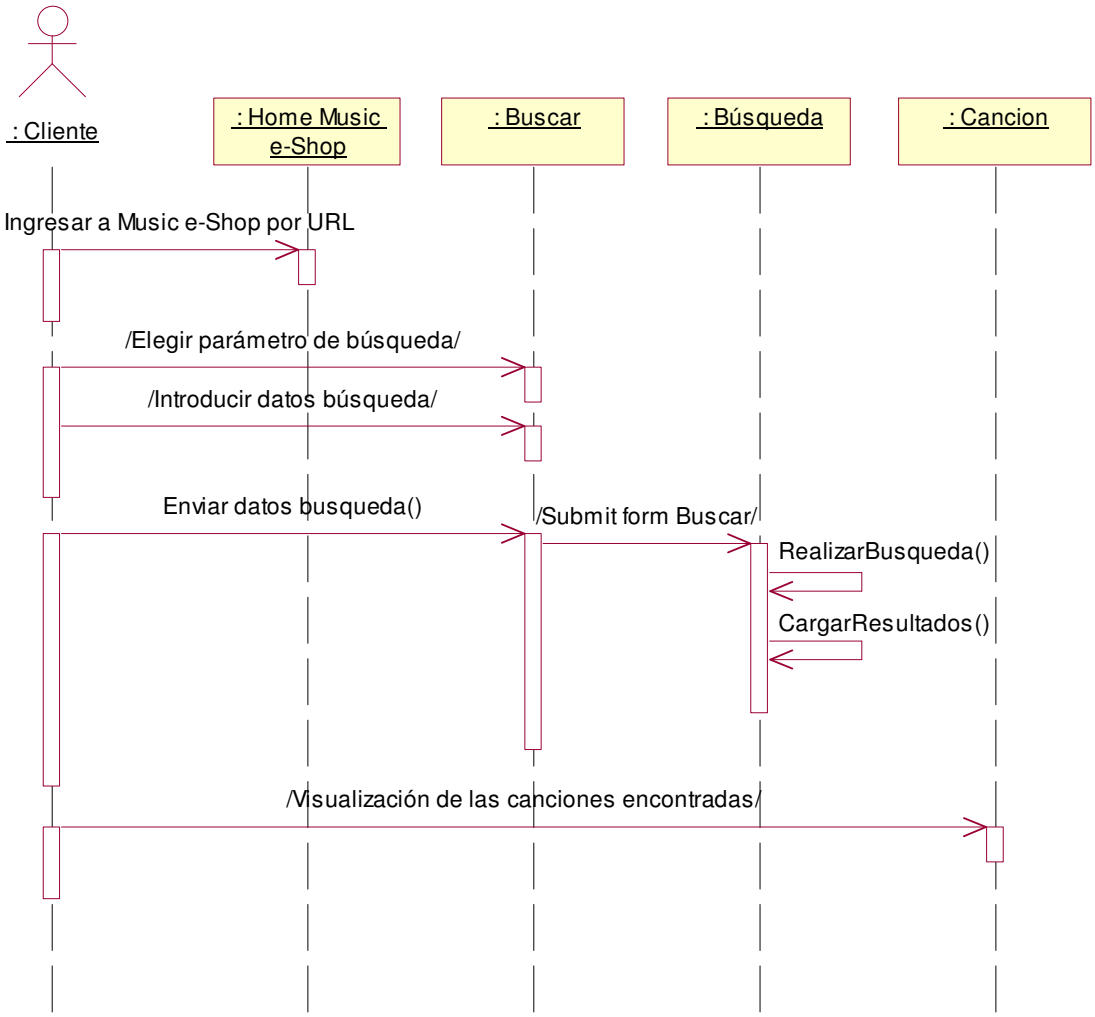

Figura 3.5: Diagrama de Secuencia para la Búsqueda de Canciones

3.3.3 Secuencia Actualización del Carrito

En la secuencia de la Figura 3.6 se muestra el flujo de las operaciones que se realizan para la gestión de las canciones que ya se encuentran agregados al carrito de compras de un Cliente. La secuencia inicia cuando un Cliente ingresa a la <<navigation class>> *Carrito*. El ingreso puede darse después de haber agregado una canción al carrito o bien, desde la <<navigation class>> *Home Music e-Shop*. Ya estando en Carrito se carga la lista de canciones ya incluidas en el carro y se realiza el cálculo de los totales

de la orden en general. A continuación el Cliente marca la o las canciones para que sean eliminadas de la orden a través del <<form>> *Seleccionar Articulo*. Por medio del <<form>> *Actualizar Carro* estos datos son enviados a la misma <<navigation class>> *Carrito*, la cual dependiendo de las canciones marcadas por el Cliente, actualiza el carrito eliminando aquellas que fueron marcadas con ese propósito. Esta tarea la realiza la operación EliminarCanciones(); y a continuación se refresca el carrito mediante la operación RefrescarCarro() para desplegar los cambios realizados. La secuencia concluye con la elección del Cliente de continuar comprando regresando al Home Music e-Shop, o bien eligiendo iniciar el proceso de pago del pedido con la Autenticación del Cliente.

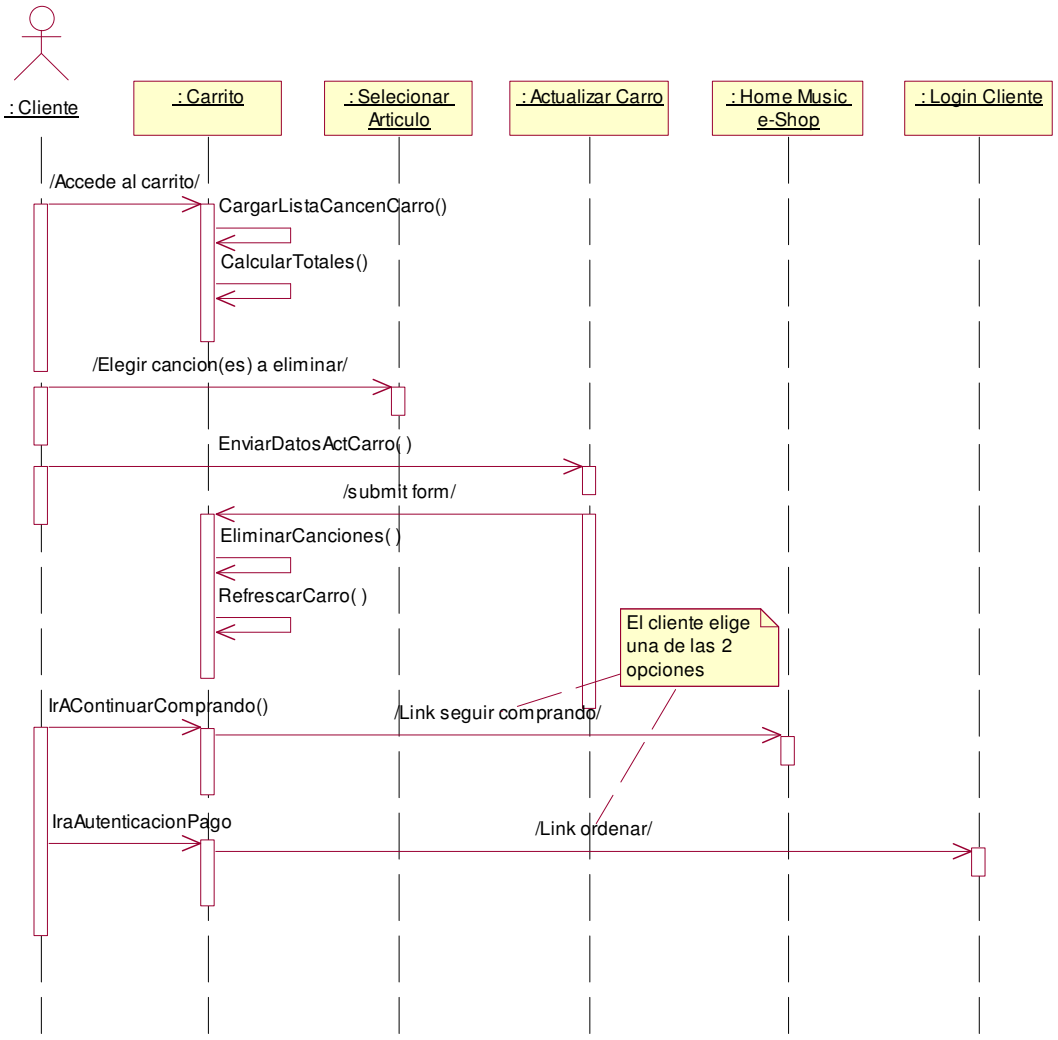

Figura 3.6: Diagrama de Secuencia para la Actualización del Carrito

3.3.4 Secuencia Autenticación de un Cliente Nuevo

Esta secuencia, que se muestra en la Figura 3.7, describe el proceso de autenticación de un Cliente nuevo. Esta autenticación incluye el registro de dicho Cliente. La secuencia inicia cuando el Cliente selecciona pasar a pagar la orden realizada desde la <<navigation class>> *Carrito*. Ya estando el control en la <<navigation class>> *Login Cliente*, el Cliente ya que no cuenta con un nombre de usuario y un password, selecciona ir a registrarse como Cliente nuevo en la <<navigation class>> *Registro Cliente*. Cuando esta <<navigation class>> se carga, lo primero que realiza es la evaluación de la tarea que va a realizarse en esa clase que, en este caso, es el registro de un nuevo Cliente. La secuencia continua con la introducción por parte del Cliente de todos y cada uno de los datos de registro requeridos por la aplicación. Esta introducción se realiza sobre el <<form>> *Datos Cliente*. Una vez que haya completado los campos requeridos, el Cliente indica al <<form>> el envió de esta información mediante la operación EnviarDatosCliente(). Los datos son enviados a la misma <<navigation class>> Registro Cliente en donde se verifica la validez de los datos proporcionados por el Cliente mediante las operaciones ValidarDatosCompletos(), ValidarCorreo() y VerificarMatchPass(), con las cuales se valida que el Cliente haya proporcionado todos los datos requerido, que el correo proporcionado por el Cliente sea válido y que el password proporcionado haya sido introducido correctamente en la forma de registro, respectivamente. En este punto se tienen dos flujos alternativos dependiendo de los resultados que arrojen dichas validaciones. Si en alguna de estas validaciones no es satisfactoria, entonces la <<navigation class>> *Registro Cliente*, debe rellenar los campos de los datos que si pasaron la comprobación y notificar al Cliente que ocurrió un error. Si todas las validaciones fueron satisfactorias, entonces Registro Cliente debe realizar las operaciones GenerarIdClienteNuevo(), RegistrarClienteNuevo() e IniciarSesionCliente(). Mediante la operación GenerarIdClienteNuevo() se genera el identificador del Cliente a registrar. RegistrarClienteNuevo() realiza el registro físico de la información del nuevo Cliente en las fuentes de datos de la aplicación. El inicio de sesión del Cliente recién registrado se realiza mediante la operación InicioSesionCliente(). Por último, la <<navigation class>> *Registro Cliente* redirecciona la navegación hacia la <<navigation class>> *Pago de Música* mediante la operación RedireccionarA() para continuar con el proceso de pago.

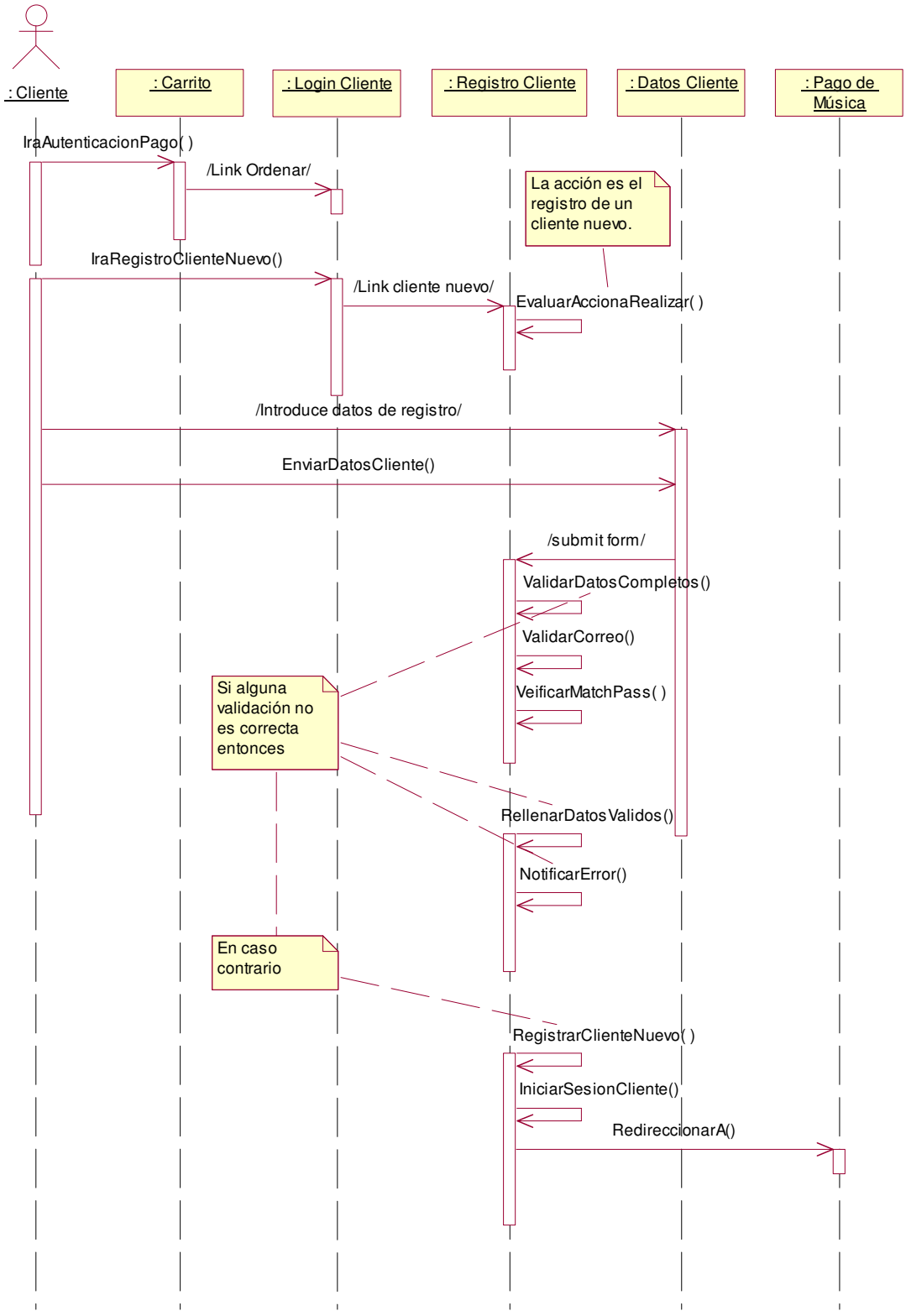

Figura 3.7: Diagrama de Secuencia para la Autenticación de un Cliente Nuevo.

3.3.5 Secuencia Autenticación de un Cliente Existente

La secuencia de la Figura 3.8 describe el proceso de autenticación de un Cliente previamente registrado. La secuencia inicia cuando el Cliente selecciona pasar a pagar la orden realizada desde la <<navigation class>> *Carrito*. Ya estando el control en la <<navigation class>> *Login Cliente*, el Cliente introduce sus datos de registro, es decir su nombre de usuario que es su correo electrónico y su contraseña. El Cliente solicita el envío de esta información mediante la operación EnviarDatosRegistro() definida en el <<form>> *Datos Registro Cliente*. La información se envía a la <<navigation class>> *Autenticación Usuario*, en donde se evalúa la acción a realizar mediante la operación EvaluarAccionaRealizar(). En esta secuencia la operación a realizar es la autenticación de un usuario Cliente. Hecho esto, la <<navigation class>> *Autenticación Usuario* realiza la autenticación del usuario utilizando los datos proporcionados mediante la operación RealizarAutenticacion(), en donde se realiza una consulta a la fuente de datos de la aplicación. Esta operación devuelve una respuesta que se evalúa en la operación EvaluarResultado(). En base a esta evaluación la secuencia puede tomar dos distintos flujos alternativos. Si la respuesta no es favorable, es decir, que los datos de registro proporcionados por el Cliente no encuentran una correspondencia en las fuentes de datos entonces se debe notificar un error, finalizando de esta manera la secuencia. Por el contrario, la respuesta es favorable, entonces *Autenticación Usuario* realiza la operación IniciarSesionCliente() con la que se inicializan las variables de sesión con los datos de registro del Cliente, finalizando la secuencia con el redireccionamiento hacia la <<navigation class>> *Pago de Música* mediante la operación RedireccionarA() para continuar con el proceso de pago.

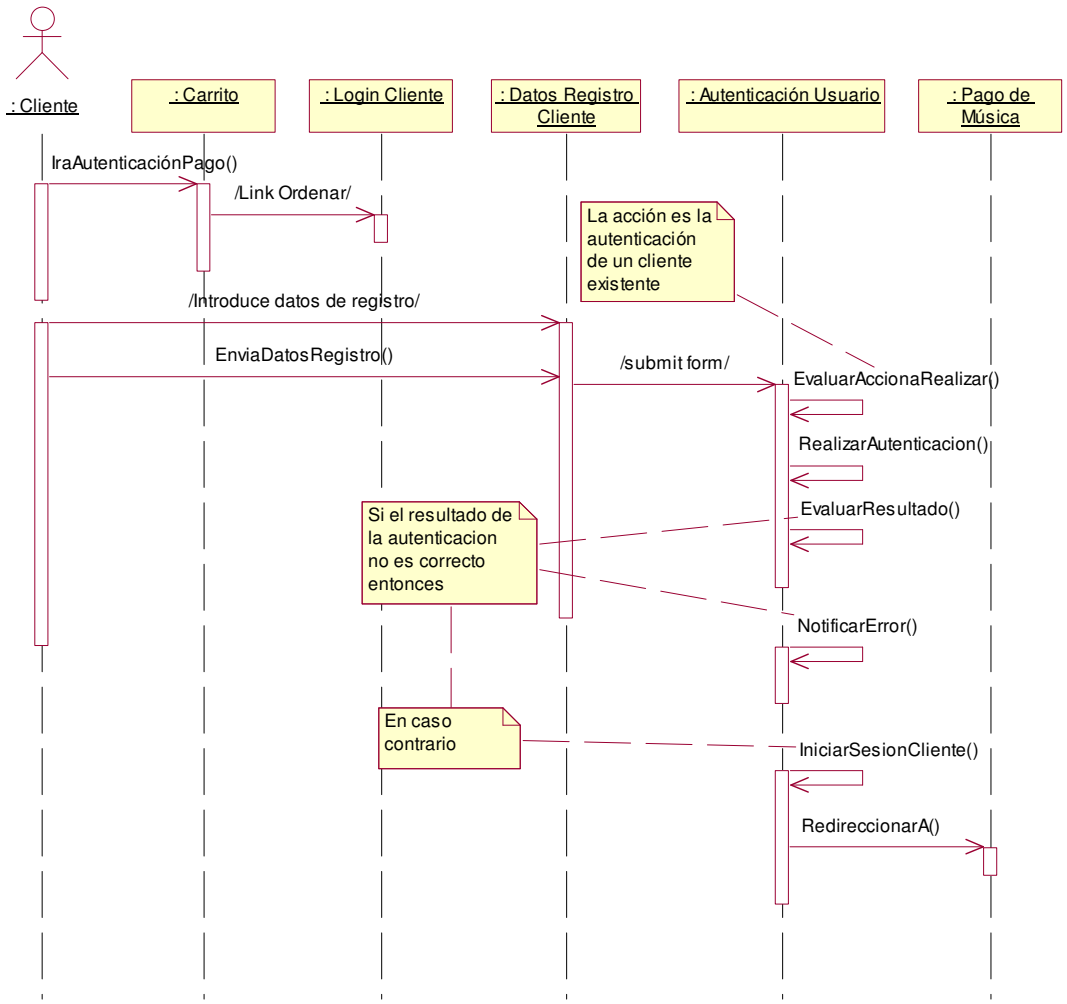

Figura 3.8: Diagrama de Secuencia para la Autenticación de un Cliente Existente.

3.3.6 Secuencia de Pago

En la Figura 3.9 se muestra la secuencia para la primera etapa del proceso de pago. En esta etapa se muestra al Cliente el resumen de la orden realizada y se le proporciona la posibilidad de cambiar los datos de facturación. La secuencia inicia una vez que el Cliente se ha validado y accede a la <<navigation class>> *Pago de Música*. Esta <<navigation class>> valida la <<navigation class>> de procedencia mediante la operación ValidarOrigen(), esto como mecanismo de seguridad. Si el origen no es válido, entonces se trata de un intento de pago malintencionado, por lo que se notifica la ocurrencia del error, y la secuencia finaliza. Si el origen es válido, entonces se continúa con el proceso cargando los datos de facturación del Cliente, los datos de la

orden y el detalle de la orden. La secuencia continua con la edición opcional de los datos de facturación del Cliente. El Cliente accede a la <<navigation class>> *Registro Cliente* desde *Pago de Música* indicando que la operación a realizar es la de modificación de los datos de facturación del Cliente. Cuando el control llega a *Registro cliente*, se evalúa la acción y de acuerdo a esta, se cargan los datos actuales del Cliente para que puedan ser modificados. El Cliente introduce la nueva información a través del <<form>> *Datos Cliente*. Esta información se reenvía a la <<navigation class>> *Registro Cliente* en donde se actualiza el registro del Cliente con los nuevos datos de facturación. Hecho esto, Registro Cliente redirecciona el control de navegación a la <<navigation class>> *Pago de Música* para continuar con el proceso de pago. Una vez de vuelta en *Pago de Música*, el Cliente debe continuar con el procedimiento de pago introduciendo la información de la tarjeta bancaria sobre la cual se debe cargar el monto de la orden. El Cliente introduce esta información mediante el <<form>> *Info Pago Música*. La información se envía en modo seguro a la <<navigation class>> *Proceso Pago Seguro* del subsistema Gestor de Transacciones para que sea este elemento quien gestione la transacción con la terminal Bancaria.

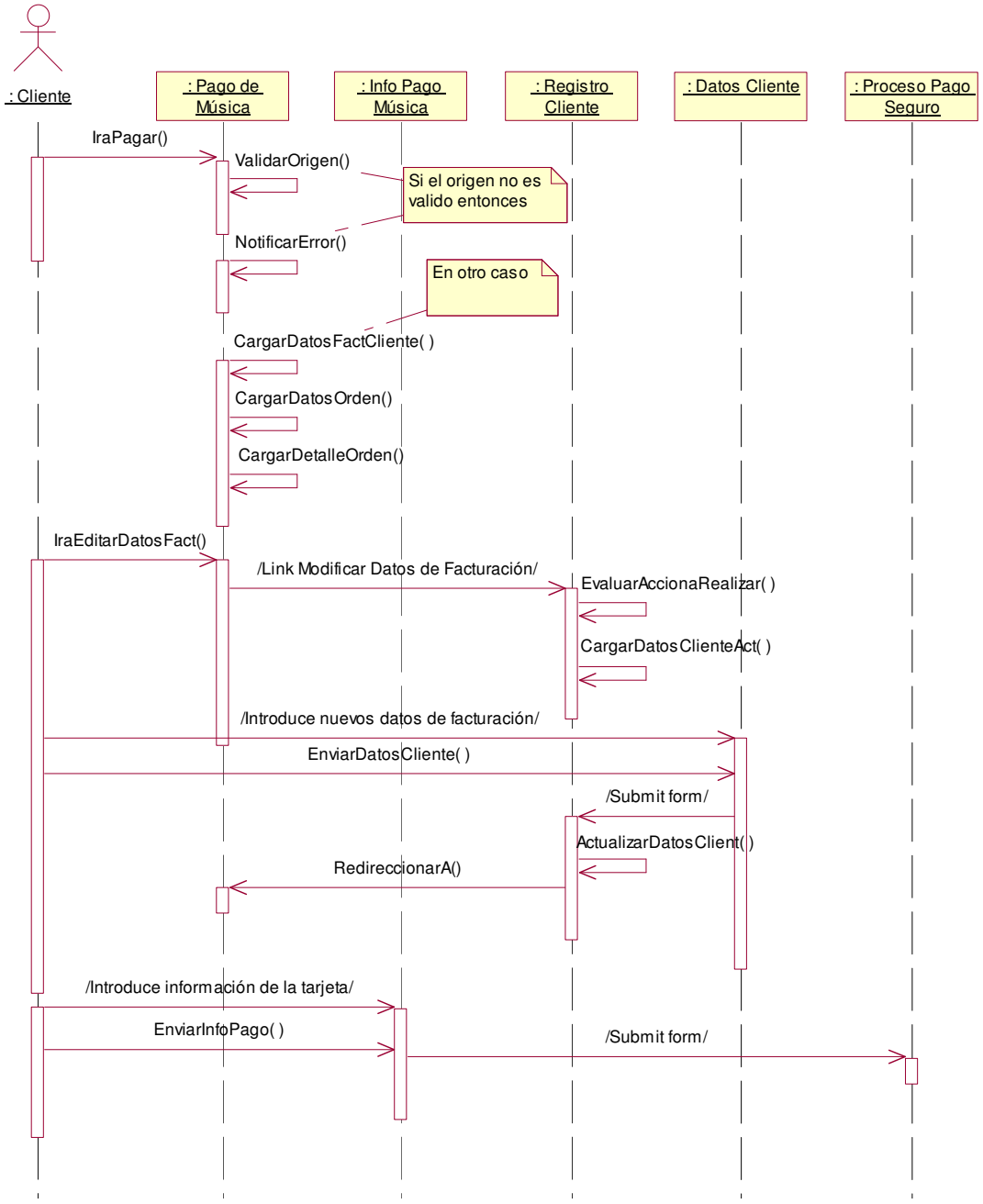

Figura 3.9: Diagrama de Secuencia para cambios en datos del cliente y captura de datos de la tarjeta de crédito.

3.3.7 Secuencia Autenticación de un Usuario Administrador

Esta secuencia, que se muestra en la Figura 3.10, describe el proceso de autenticación de un usuario Administrador para acceder al Home Administración. La secuencia inicia cuando el usuario Administrador accede a la <<navigation class>> *Login Admon* mediante direccionamiento URL. Una vez en *Login Admon*, el Administrador debe

introducir sus datos de acceso por medio del <<form>> *Datos Registro Admon*. Esta información es enviada hacia la <<navigation class>> *Autenticación Usuario*. Cuando en *Autenticación Usuario* se recibe esta información se evalúa la acción a realizar, en este caso se trata de la autenticación de un usuario del tipo Administrador. Hecho esto la <<navigation class>> *Autenticación Usuario* realiza la autenticación del usuario Administrador, la cual devuelve un resultado. Si el resultado no es favorable, entonces se debe notificar un error finalizando de esta manera la secuencia. Si el resultado es favorable, entonces *Autenticación Usuario* realiza la operación IniciarSesionAdmon() con la que se inicializan las variables de sesión con los datos de registro del Administrador, finalizando la secuencia redireccionando la navegación hacia la <<navigation class>> *Home Administración* mediante la operación RedireccionarA() en donde se proporcionan al Administrador diversas opciones para operación de Music e-Shop.

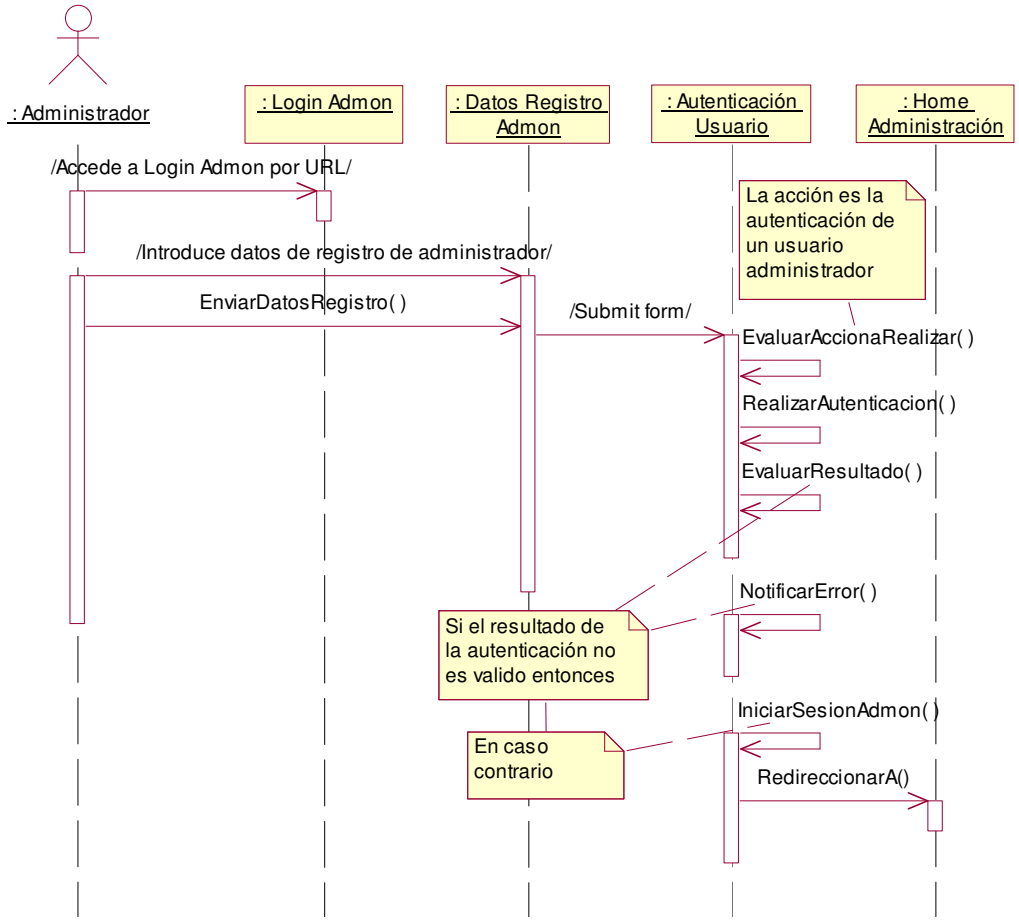

Figura 3.10: Diagrama de Secuencia para la Autenticación de un Usuario Administrador.

3.3.8 Secuencia Adición de un Género Nuevo al Catálogo

Esta secuencia, que se muestra en la Figura 3.11, describe el procedimiento de adición de un nuevo género al catálogo de canciones de la tienda por parte del usuario Administrador, administrador de Music e-Shop. La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración*, una vez que se ha validado de manera correcta como un usuario válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogo*. En esta <<navigation class>> se carga la lista del catálogo con todos los géneros actuales de Music e-Shop. Para agregar una nuevo género, el usuario Administrador lanza la operación IraRegistroGeneroNuevo() para navegar hacia la <<navigation class>> *Registro Género*. Cuando *Registro Género* se carga, se evalúa la acción a realizar que, en este caso, es la agregación de un nuevo género al catálogo. El usuario Administrador, a través del <<form>> *Datos Género,* introduce la información del nuevo género a agregar. El Administrador incluye, opcionalmente, un archivo de una imagen para representar al nuevo genero en el catálogo e independientemente de haber incluido o no el archivo, el usuario lanza la operación EnviarDatoGenero(). Esta información se reenvía hacia la <<navigation class>> *Registro Género* en donde se valida que dicha información sea la apropiada y, en base a esta evaluación, la secuencia prosigue en cualquiera de dos flujos alternativos. Si la validación no es satisfactoria entonces se notifica el error y la secuencia termina. Por el contrario, si la validación de la información es correcta, la secuencia continua en la <<navigation class>> *Registro Género* con la operación RegistrarGenNuevo() con la que se efectúa el registro del nuevo género en la fuente de datos de Music e-Shop. El género recién creado no contiene ninguna canción; las canciones se agregarán al género cuando se lleve a cabo el registro de la canción.

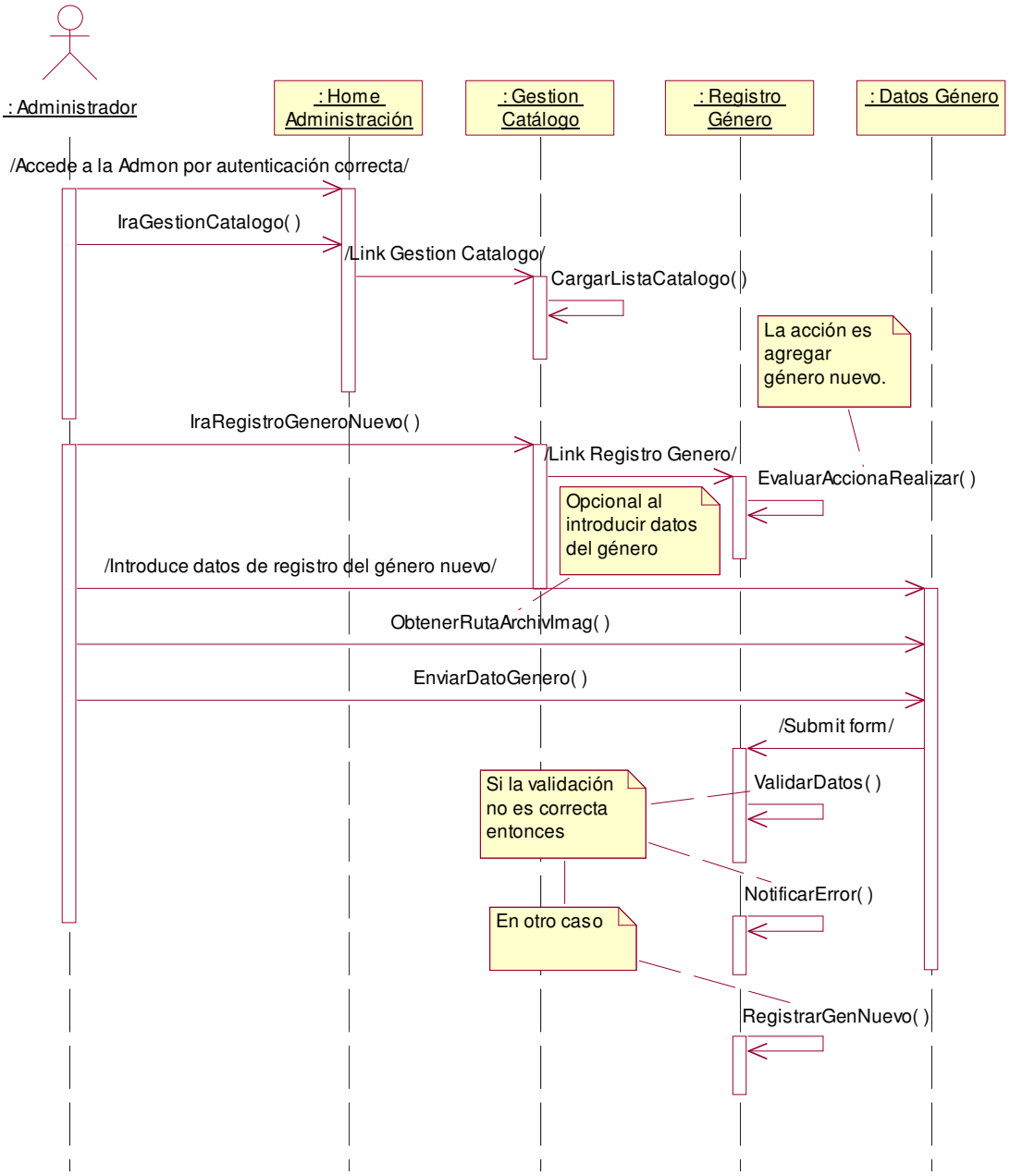

Figura 3.11: Diagrama de Secuencia para la Adición de un Género Nuevo al Catálogo.

3.3.9 Secuencia Edición de un Género Existente

Esta secuencia, que se muestra en la Figura 3.12, describe la edición de un género ya existente en el catálogo de canciones por parte del usuario Administrador de Music e-Shop. La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración*, una vez que se ha validado de manera correcta como un usuario

válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogo*. En esta <<navigation class>> se carga la lista del catálogo con todos los géneros actuales de la tienda. Este listado es el <<index>> *Catálogo*. De este <<index>> de géneros, el Administrador selecciona a uno de ellos para editarlo mediante la operación Editar() asociada a dicho género. Esta operación hace navegar al Administrador a la <<navigation class>> *Registro Género*. Cuando *Registro Género* se carga se evalúa la acción a realizar que, en este caso, es la edición de un género ya existente en el catálogo. Subsecuentemente, se cargan los datos actuales del género a editar en el <<form>> *Datos Género*. Una vez que los datos del género aparecen cargados en el <<form>>, el usuario Administrador modifica la información del género y , una vez hecho los cambios deseados, lanza la operación EnviarDatosGénero(). Esta información se reenvía hacia la <<navigation class>> *Registro Género* en donde se valida que dicha información sea apropiada y, en base a esta evaluación, la secuencia prosigue en uno de dos flujos alternativos. Si la validación no es satisfactoria entonces se notifica el error y la secuencia termina. Por el contrario, si la validación de la información es correcta, la secuencia concluye con la operación ActualizarGen() de la <<navigation class>> *Registro Género*, con la cual se lleva a cabo la actualización del registro en la fuente de datos del género editado. Cabe señalar que en la implementación, si un género tiene canciones asociados a el, la edición del mismo debe incluir la persistencia de este vinculo aun cuando el género sea editado.

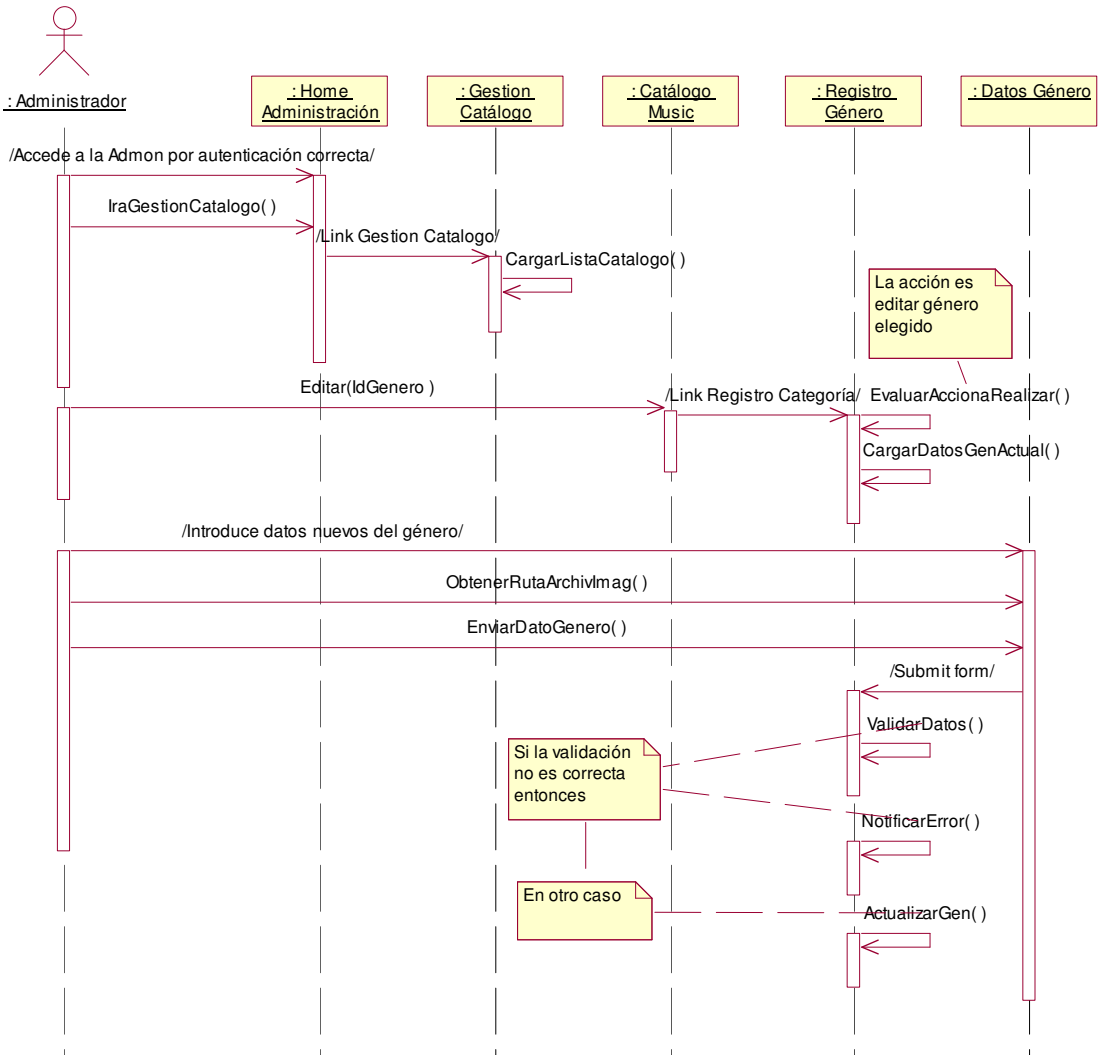

Figura 3.12: Diagrama de Secuencia para la Edición de un Género Existente.

3.3.10 Secuencia Eliminación de un Género Existente

La eliminación de un género existente en el catálogo de canciones, llevada a cabo por el administrador, se describe en esta secuencia (Figura 3.13). La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración* una vez que se ha validado de manera correcta como un usuario válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogos*. En esta <<navigation class>> se carga la lista del catálogo con todos los géneros actuales de la tienda. Este listado es el <<index>> *Catálogo*. De este <<index>> de géneros, el Administrador selecciona a uno de ellos para eliminarlo mediante la operación Eliminar() asociada a dicho género. Esta operación hace

navegar al Administrador a la <<navigation class>> *Registro Género*. Cuando *Registro Género* se carga, se evalúa la acción a realizar que, en este caso, es la eliminación de un género ya existente en el catálogo. La secuencia concluye con la operación EliminarGen() de la <<navigation class>> *Registro Género*, con la cual se lleva a cabo la eliminación (marcada) del registro en la fuente de datos del género editado. Cabe señalar que en la implementación, si un género tiene canciones asociadas a él, dichas canciones deben moverse a un género general preexistente que no pueda ser eliminado.

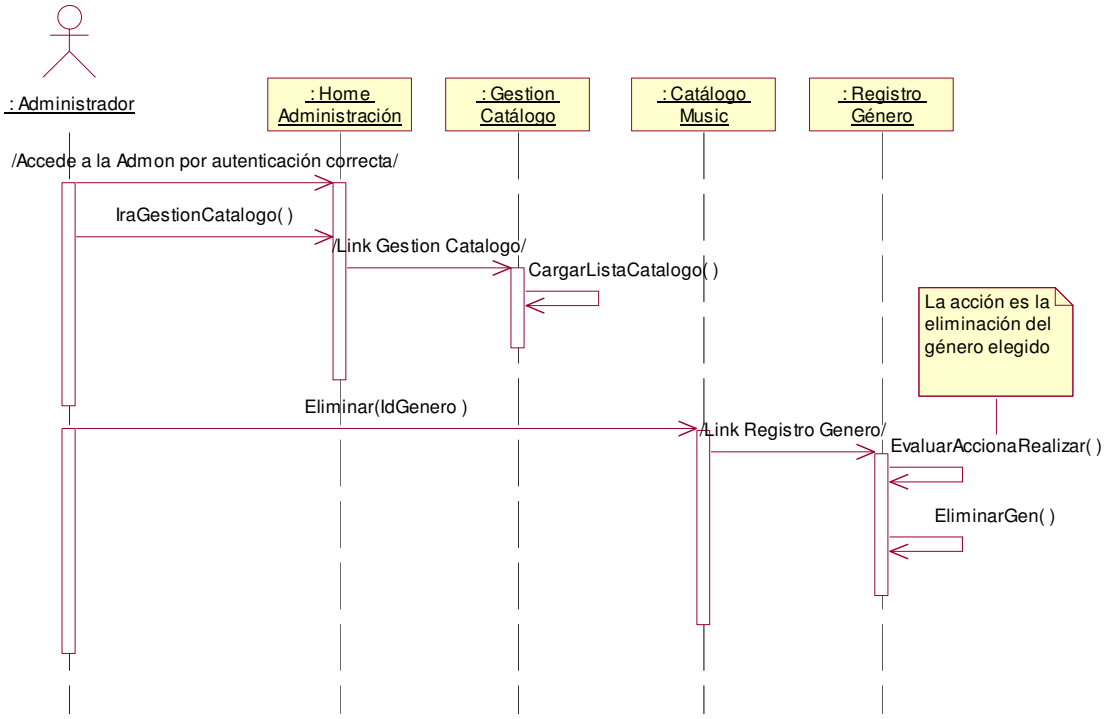

Figura 3.13: Diagrama de Secuencia para la Eliminación de un Género Existente.

3.3.11 Secuencia Adición de una Canción Nueva al Catálogo

Esta secuencia, que se muestra en la Figura 3.14, describe el procedimiento de adición de una nueva canción al catálogo de canciones de la tienda por parte del usuario Administrador. La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración* una vez que se ha validado de manera correcta como un usuario válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogo*. En esta <<navigation class>> se carga la lista del catálogo con todos los géneros actuales de la tienda. Para agregar una nueva

canción, el usuario Administrador lanza la operación IraRegistroCancionNueva() para navegar hacia la <<navigation class>> *Registro Canción*. Cuando *Registro Canción* se carga, evalúa la acción a realizar que, en este caso, es la agregación de una nueva canción al catálogo. El usuario Administrador, a través de los campos definidos en el <<form>> *Datos Canción*, introduce los datos de la nueva canción a agregar. Entre los datos requeridos para el registro se encuentra la descripción de la canción, el precio, género a la cual se va a asociar, fecha de publicación y archivo de la canción que se esta registrando. El Administrador introduce los datos de registro de una nueva canción y la secuencia del registro continúa, cuando el usuario Administrador lanza la operación EnviarDatosCanc() del <<form>> *Datos Canción*. Esta información es enviada hacia la <<navigation class>> *Registro Canción* en donde se valida que dicha información sea correcta. Si la validación no es satisfactoria entonces se notifica el error y la secuencia termina. Por el contrario, si la validación de la información es correcta, la secuencia concluye con la operación RegistrarCancionNueva() con la que se efectúa el registro de la nueva canción en la fuente de datos de Music e-Shop.

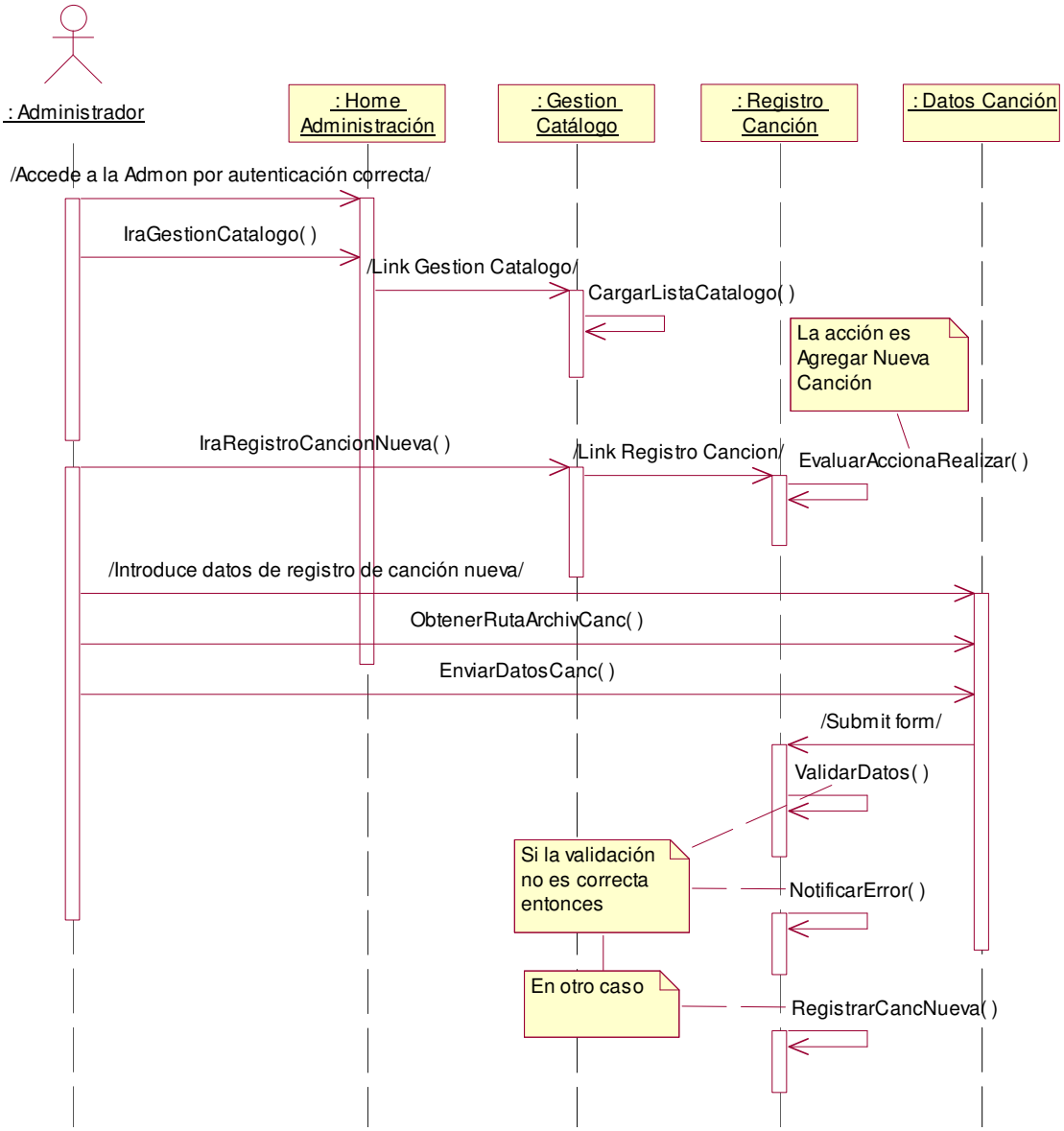

Figura 3.14: Diagrama de Secuencia para la Adición de una Canción Nueva al Catálogo.

3.3.12 Secuencia Edición de una Canción Existente

La secuencia que se muestra en la Figura 3.15 describe el procedimiento de edición de una canción ya registrada en el catálogo de canciones de la tienda. La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración* una vez que se ha validado de manera correcta como un usuario válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogo*. En esta <<navigation class>> se carga el <<index>> *Catálogo*, que es un

listado de todos los géneros existentes en Music e-Shop. De este listado, el usuario Administrador selecciona, mediante la operación VerCancionesdeGen(), a uno de ellos para ver el listado de canciones que contiene. Al hacer esto se carga el <<index>> *Género*. Este <<index>> es un listado de las canciones que contiene el género seleccionado. De este listado el usuario Administrador selecciona, mediante la operación Editar() asociada a dicho género, a la canción el cual va a ser editada. Esta operación hace navegar al Administrador a la <<navigation class>> *Registro Canción*. Cuando *Registro Canción* se carga se evalúa la acción a realizar que, en este caso, es la edición de una canción ya existente en el catálogo. En seguida, se cargan los datos actuales de la canción a editar en el <<form>> *Datos Canción*. Una vez que los datos de la canción aparecen cargados en el <<form>>, el usuario Administrador realiza los cambios deseados en el registro de la canción. Una vez que ha introducido los nuevos datos, el Administrador lanza la operación EnviarDatosCancion() del <<form>> *Datos Canción*. Esta información se envía hacia la <<navigation class>> *Registro Canción* en donde se valida que dicha información sea adecuada y, en base a esta evaluación, la secuencia prosigue en uno de dos flujos alternativos. Si la validación no es satisfactoria entonces se notifica el error y la secuencia termina. Por el contrario, si la validación de la información es correcta, la secuencia concluye con la operación ActualizarCancion() con la que se efectúa la actualización del registro de la canción en la fuente de datos de Music e-Shop.

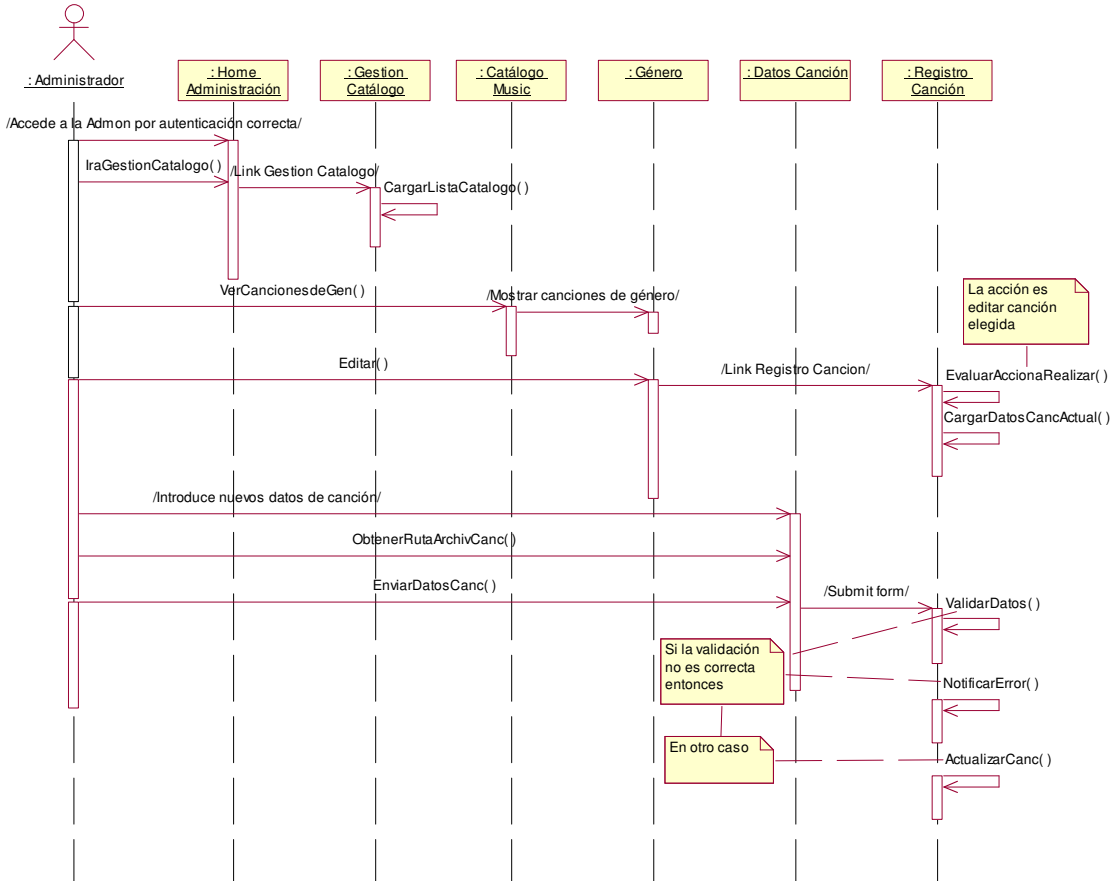

Figura 3.15: Diagrama de Secuencia para la Edición de una Canción Existente.

3.3.13 Secuencia Eliminación de una Canción Existente

La secuencia para el procedimiento de eliminación de una canción del catálogo de Music e-Shop se describe en la Figura 3.16. La secuencia inicia cuando el Administrador accede a la <<navigation class>> *Home Administración* previa autenticación correcta como un usuario válido. Desde *Home Administración* el Administrador navega hacia la <<navigation class>> *Gestión Catálogos*. Esta <<navigation class>> carga el <<index>> *Catálogo*, que es un listado de todos los géneros existentes en Music e-Shop. De este listado, el usuario Administrador selecciona, mediante la operación VerCancionesdeGenero(), a uno de estos géneros para ver el listado de canciones que contiene. Al hacer esto se carga el <<index>> *Género*. Este <<index>> es un listado de las canciones que contiene el género seleccionado. La secuencia prosigue cuando el usuario Administrador selecciona a una de ellas para eliminarla mediante la operación Eliminar() asociada a dicha canción. Esta

operación hace navegar al Administrador a la <<navigation class>> *Registro Canción*. Cuando *Registro Canción* se carga se evalúa la acción a realizar que, en este caso, es la eliminación de una canción. La secuencia concluye con la operación EliminarCanc() de la <<navigation class>> *Registro Canción*, con la cual se lleva a cabo la eliminación (marcada) del registro en la fuente de datos de la canción seleccionada.

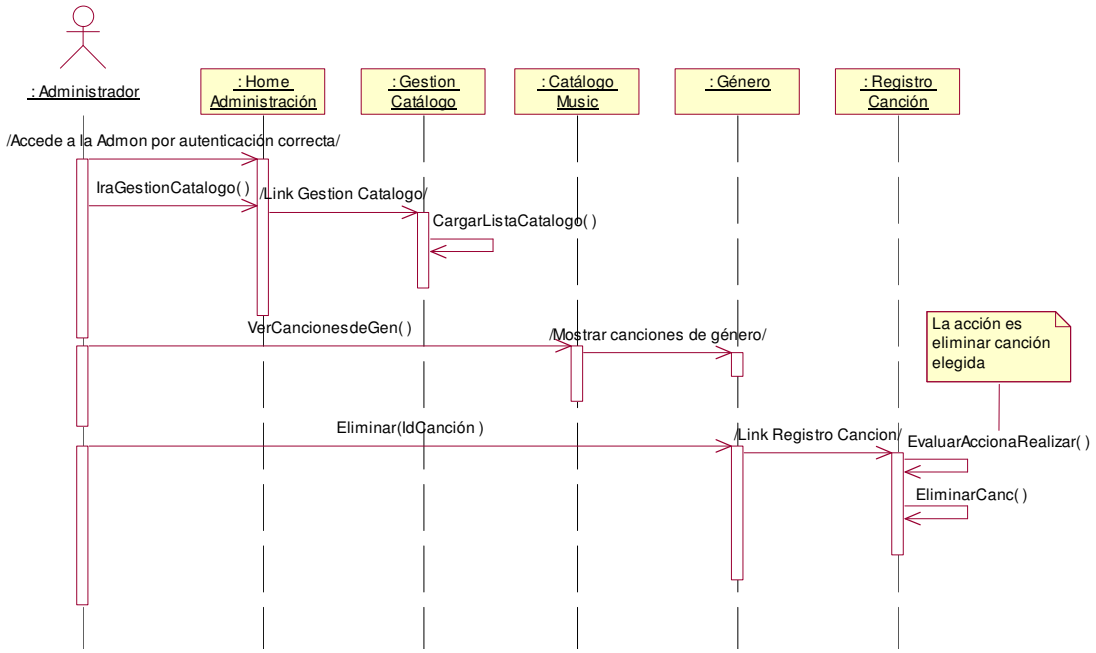

Figura 3.16: Diagrama de Secuencia para la Eliminación de una Canción Existente.

3.4 Escenarios Web

Los diagramas UML que se utilizan para visualizar los escenarios de navegación son los diagramas de transición de estados para dar a conocer la descripción de situaciones de uso de un sitio Web. Para el caso de Music e-Shop se han modelado los escenarios Web que describen las acciones necesarias de funcionalidades provistas por tal aplicación Web.

3.4.1 Escenario Music e-Shop

En la Figura 3.17 se aprecia el escenario que muestra la operación de la tienda virtual de manera general para el proceso de compra. Cuando un usuario accede al Home Music e-Shop se carga el listado de géneros y el formulario de búsqueda. En esta

situación el Cliente puede seleccionar un género o bien intentar una búsqueda. Si el Cliente intenta una búsqueda entonces, la situación cambia a Procesando Búsqueda. Cuando la búsqueda concluye, entonces se despliega la lista de los resultados, de donde el Cliente puede seleccionar una o más canciones para agregarlas al carrito. Ahora bien, si desde el Home Music e-Shop el Cliente selecciona un género, entonces se presenta el listado de canciones del género seleccionado, donde puede decidir agregar la o las canciones que quiera al carrito de compras. Ya estando en el carrito, el Cliente puede elegir seguir comprando regresando a Home Tienda para seleccionar otra canción y adicionarla a la orden, o realizar cambios sobre las canciones que ya se encuentran en el carrito del Cliente. Por último, estando en el carrito, un Cliente puede decidir comprar las canciones que contenga, entonces la situación cambia para iniciar el proceso de pago en Login Cliente. Si el Cliente es nuevo entonces, la situación cambia para realizar que el Cliente sea registrado. Cuando el registro se realiza de manera correcta se inicia el proceso de pago en su primera etapa. Cuando el Cliente introduce sus datos de registro, entonces se lleva a cabo la autenticación de usuario. Si es exitosa se inicia el proceso de pago en su primera etapa, de lo contrario se retoma la situación para que el Cliente vuelva a intentar autenticarse de manera correcta. Estando en la situación de la primera etapa del pago, el Cliente puede elegir modificar los datos de facturación. En esta situación, el Cliente introduce los datos de la tarjeta con la cual desea efectuar el pago. Una vez que el Cliente acepta el cargo a la tarjeta, la información de la tarjeta es enviada para que se realice la transacción. La situación cambia entonces para procesar dicha operación. El proceso puede terminar con dos resultados. Si el resultado es exitoso, entonces la situación final es de confirmación de la operación efectuada, en la cual el Cliente puede imprimir la factura y descargar las canciones de su compra. Por el contrario, si el resultado del pago intentado no es exitoso, entonces la situación final es la notificación del error ocurrido.

En este escenario también se presenta la situación que representa al proceso de autenticación correcta de un usuario tipo Administrador. En este caso el usuario Administrador accede directamente, sin pasar por Home Music e-Shop, a la interface de ingreso de la sección de Administración de Music e-Shop en donde el Administrador introduce los datos de registro. El proceso de autenticación se representa con una situación, después de la cual, en base al resultado de dicho proceso, se transita hacia el subsistema de Administración de Music e-Shop o de nuevo a la interface de ingreso.

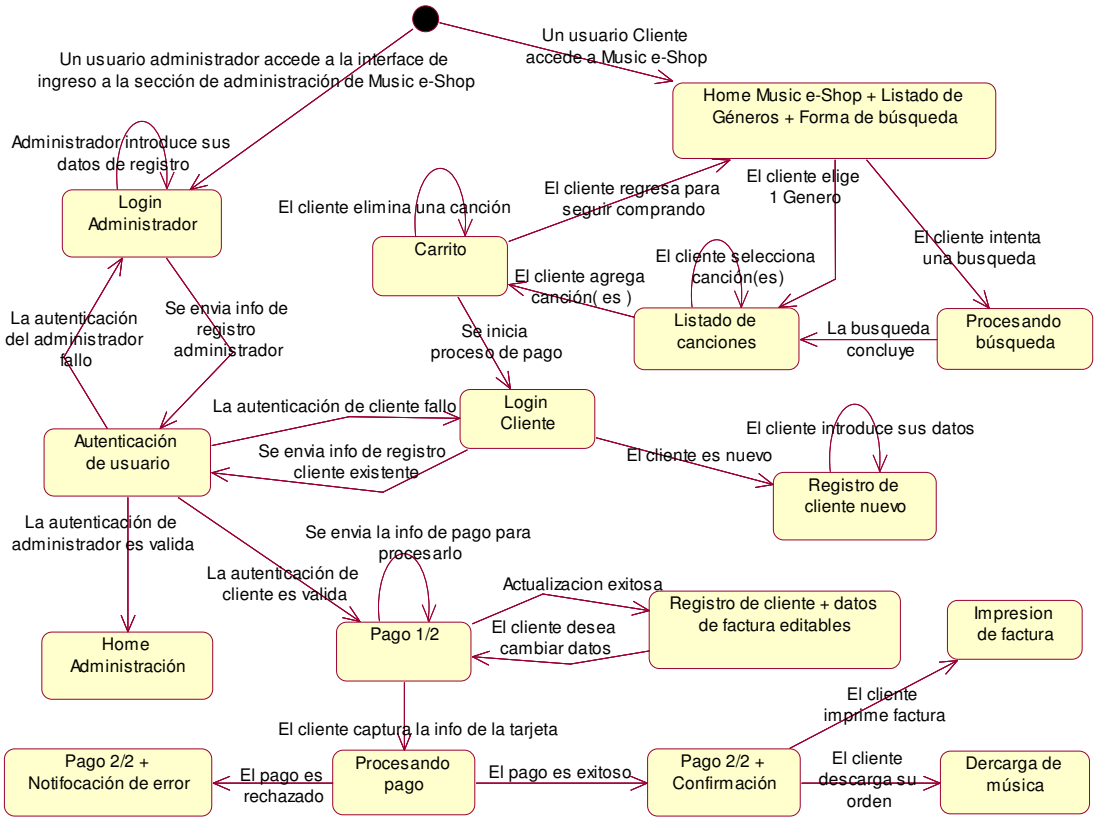

Figura 3.17: Escenario Web Music e-Shop, parte 1.

3.4.2 Escenario Administración de Music e-Shop

En esta segunda parte del escenario Web Music e-Shop, que se muestra en la Figura 3.18, referente a la Administración de la tienda on-line, se describen las situaciones asociadas con la funcionalidad de la visualización de informes y gestión del catálogo. Una vez que un usuario Administrador accede por autenticación correcta a la Administración de Music e-Shop, la situación va a depender de la selección que haga el Administrador en base a la funcionalidad que desea utilizar. En el caso que el Administrador selecciona operar sobre la visualización de informes de la tienda, se carga el módulo correspondiente. La visualización de informes se describe con tres situaciones, la presentación de la funcionalidad, el despliegue del listado de colección de informes y la visualización de un informe específico. Mediante estas tres situaciones el Administrador puede visualizar un informe de la colección de informes disponibles para la tienda.

Por otro lado, la funcionalidad de la gestión del catálogo presenta varias situaciones. Cuando se accede a este módulo de la aplicación se presenta un listado de las categorías que se incluyen en el catálogo. En esta situación, el Administrador puede seleccionar registrar un género nuevo, registrar una canción nueva, o bien, seleccionar a un género del listado para visualizar las canciones asociadas a dicho género o bien, para editarlo o eliminarlo. Cuando se selecciona ver el listado de canciones de un género, las opciones para el usuario Administrador en esa situación son cuatro; el usuario puede seleccionar a una canción del listado para editarla o eliminarla. Cada una de estas funcionalidades presenta distintas situaciones, las cuales se presentan de acuerdo a las operaciones que el Administrador realice en la aplicación.

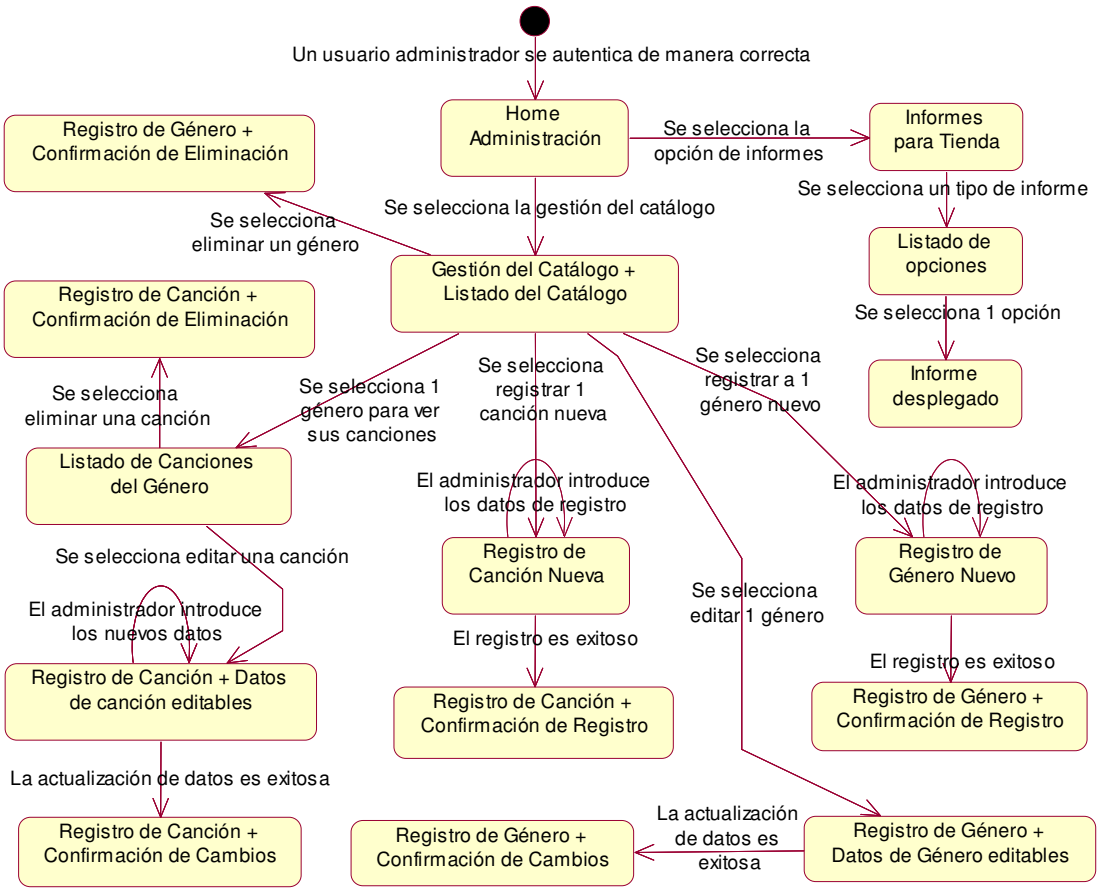

Figura 3.18: Escenario Web Music e-Shop, parte 2.

Capítulo 4

Implementación de Music e-Shop

En este capítulo se presenta la ultima parte del presente proyecto, la implementación de Music e-Shop basada en el modelado UWE elaborados en el capítulo 3 y 4 de este documento. En la primera sección se presentan las herramientas de desarrollo empleadas en la implementación de Music e-Shop. En la segunda sección se presenta la implantación de XML en el sistema. Por ultimo, en la tercera se muestra la aplicación en acción, es decir, se muestra a Music e-Shop en marcha mediante un ejemplo de su funcionamiento.

4.1 Herramientas de Desarrollo

El proceso de desarrollo de una aplicación Web, en lo que respecta a la implementación, depende en mucho de la adecuada elección del lenguaje de programación, servidor de bases de datos y servidor Web. En esta sección se presenta una descripción breve de las herramientas de software empleadas en el desarrollo de la aplicación Web Music e-Shop. También se subraya el concepto del Formato Estructurado de Texto debido a que también se va a emplear como una herramienta en el desarrollo de la aplicación.

4.1.1 Servlets Java

Java, desarrollado por SUN Microsystems a principios de los años 90, es un lenguaje de programación de propósito general, orientado a objetos puro, distribuido, interpretado, robusto, seguro, multi-plataforma, de alto desempeño, productivo, multi-hilos y dinámico. Java dispone de muchas utilidades que lo hacen idóneo para la web. Los servlets heredan todas las ventajas y utilidades de Java. Un Servlet es un programa Java que se ejecuta en un servidor Web. Los clientes pueden invocar un servlet utilizando el protocolo HTTP.

Un servlet acepta peticiones de un cliente, procesa la información relativa a la petición realizada por el cliente y le devuelve a este los resultados qhe podrán ser mostrados

mediante applets, paginas HTML, etc. Algunas de las características que ofrecen los servlets según [Ceballos03] son:

- Al estar escritos en Java, son independientes de la plataforma.
- Consumen menos recursos porque solo son cargados la primera vez que se solicitan sus servicios. Las siguientes peticiones crearan hilos de ejecución.
- Son mas rápidos que los programas CGI y que los scripts porque, por un lado se precompilan y, por otro, no se tiene que generar un nuevo proceso cada vez que se invocan.
- Son seguros y portables debido: 1) a que se ejecutan bajo la maquina virtual de Java, 2) al mecanismo de excepciones de Java, y 3) al uso del administrador de seguridad de Java(Java Security Manager).
- No requieren soporte para Java en el explorador del cliente, ya que operan en el dominio del servidor y envían los resultados en HTML. No obstante, se puede utilizar otras interfaces de cliente como aplicaciones Java o applets.

4.1.2 MySQL

MySQL es un sistema gestor de bases de datos (SQL), con un rendimiento excelente, desarrollado bajo la filosofía de código abierto. Características: capacidad de manejar un extenso subconjunto del lenguaje SQL estándar, alta disponibilidad de plataformas y sistemas, proporciona conectividad confiable y segura, permite la ejecución de transacciones y el manejo de claves foráneas, permite replicación y provee la capacidad de realizar búsquedas e indexaciones de campos de texto, entre otras características más. Para consultar esta información y obtener mas detalle sobre MySQL acceder a la página oficial: dev.mysql.com.

4.1.3 Tomcat

Tomcat, también llamado Jakarta Tomcat o Apache Tomcat, funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de Java Servlet y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web autónomo con soporte de servlets y JSP, el cual puede ser usado en entornos con alto nivel de tráfico y alta disponibilidad. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. Debido a que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la Máquina Virtual Java (JVM: por sus siglas en ingles). Tomcat impulsa numerosas aplicaciones Web de larga escala y misión critica a través de un diverso rango de industrias y organizaciones. Así, Tomcat es un producto muy robusto, altamente eficiente y uno de los más potentes contenedores de Servlets existentes. Para consultar esta información y obtener mas detalle de Tomcat acceder a la página oficial: tomcat.apache.org/.

4.1.4 eXtensible Markup Language

XML es un estándar con independencia del tipo de implementación seleccionado. Es decir que podemos usar herramientas de distintos proveedores para estructurar datos con la especificación XML, almacenarlos en una base de datos, realizar búsquedas o ejecutar cualquier proceso. Así los datos serán accesibles y procesables por todas las herramientas que siguen el estándar XML independientemente de su plataforma y su fabricante.

4.2 Sirviendo XML en el Sistema Music e-Shop

La operación de datos a través de la Web es uno de los medios más convenientes para realizar transacciones. Los lenguajes que permiten efectuar este tipo de interfaz entre un sitio Web y el usuario necesitan del uso imprescindible de bases de datos ya que sin ellas no hay forma de poder manejar la información que interviene en cualquier tipo de transacción.

Cuando se quieren enviar datos de una página a un servidor para que estos sean alojados para su consulta posterior, se deben tratar los conceptos de formularios, bases de datos, registros, etc. El hecho real de conectarse a una base de datos para obtener información mediante la web, hace a veces que un servidor llegue a saturarse de tal forma que es improbable acceder al mismo; todo esto por que se generan sinfín de conexiones a una base de datos alojada en un servidor. XML puede solucionar este problema alojando en un documento XML la información de una base de datos y

permitir así la obtención de información sin que esto requiera de generar conexiones que saturen el servidor.

XML es un lenguaje que permite crear su propia estructura de datos, lo cual facilita declaraciones de contenido más correctas y resultados de búsquedas más precisas en distintas plataformas, ofrece la manipulación de datos basados en Web, también garantiza que los datos sean uniformes e independientes de aplicaciones o fabricantes.

Ahora, parte importante del objetivo del presente trabajo es desarrollar una aplicación que permita guardar en un archivo XML los datos extraídos de una base de datos. También, permita leer dichos archivos XML y analizarlos, así como mostrar los datos contenidos en aquellos archivos en formato HTML, como se muestra en la Figura 4.1.

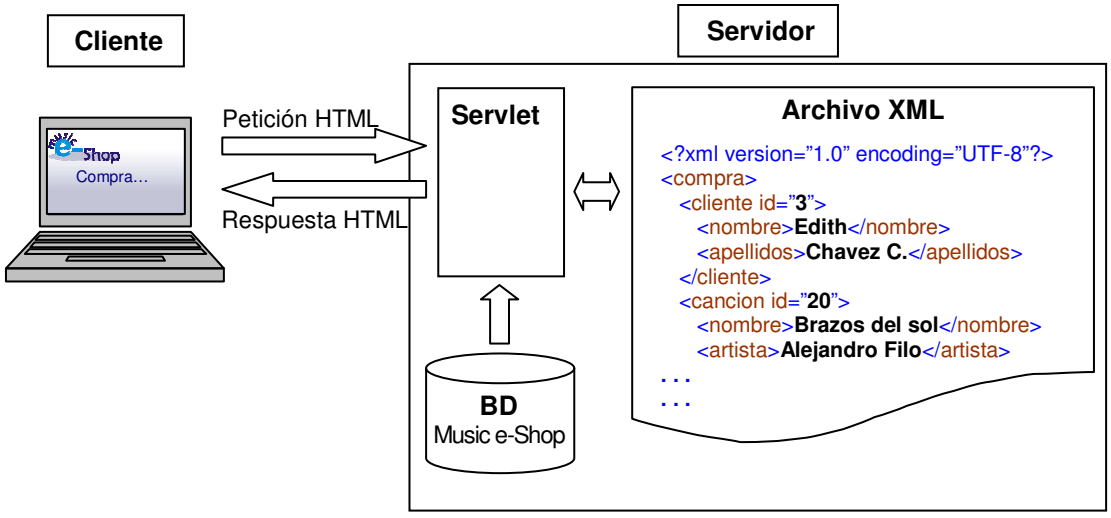

Figura 4.1: Esquema para mostrar una compra usando Servlets y XML.

La idea para cubrir el cometido es la siguiente: una vez llevada a cabo la compra de un producto por parte del usuario y almacenados los datos de dicha compra en la base de datos, se utiliza esta información para emitir los comprobantes de la compra correspondientes de la base de datos de la aplicación e-commerce. La escritura del archivo en formato XML con los datos de una compra, es llevada a cabo por un servlet. El archivo XML relacionado con su correspondiente Schema y hoja de estilos (XSL), es analizado por medio de un analizador sintáctico o parser, para generar un comprobante de la compra en formato HTML; que otro servlet devuelve a través del HTTP.

El primer punto a tratar es la elaboración del archivo XML, con su correspondiente Schema y hoja de estilos. Como se había mencionado anteriormente el archivo XML se genera por medio de un servlet, así que en la Figura 4.2 se muestra el fragmento de

código que construye a dicho archivo. En el código se puede observar que se crea el archivoXML y se llama a los métodos cargarDatosCliente() y cargarDatosOrden() los cuales hacen acceso a la base de datos para obtener los datos del cliente y la orden respectivamente para escribirlos en el documento XML.

```
/**** Creacion del archivo XML con los datos de la compra ******/
    try{
       //Crear el directorio del cliente si dicho directorio no existe.
        File fich=new File(cliente);
        fich.mkdir();
        //Abrir el fichero para la compra.
        fw=new FileWriter(cliente+"/"+numeroVenta+".xml", false);
        fichCompra=new PrintWriter(fw);
        //Escribe el contenido del archivo XML
        fichCompra.println("<?xml version='1.0'?>\n"+
            "<?xml-stylesheet type='text/xsl' href='../compra.xsl'?>\n"+
            "<compra xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'\n"+
            "xsi:noNamespaceSchemaLocation='../compra.xsd'>\n");
        fichCompra.println("<fecha>"+fecha+"</fecha>\n<hora>"+hora+"</hora>\n");
        cargarDatosCliente();
        cargarDatosOrden();
        fichCompra.println("</compra>\n");
        fichCompra.close();
        fw.close();
    }
    catch(IOException e){ out.println(e.getMessage()); }
```

Figura 4.2: Código para la creación del archivo XML.

El resultado de la creación del archivo XML utilizando un servlet se puede observar en la Figura 4.5.

```
<?xml version='1.0'?>
<?xml-stylesheet type='text/xsl' href='../compra.xsl'?>
<compra xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='../compra.xsd'>
    <fecha>3/2/2008</fecha>
    <hora>19:40:49</hora>
    <cliente email='edith_ch@hotmail.com'>
        <nombre>Edith </nombre>
        <apellidos>Chavez Contreras</apellidos>
        <direccion>Av. Juarez no. 36 Col. Chapultepec</direccion>
        <ciudad>Puebla</ciudad>
        <estado>Puebla</estado>
        <pais>Mexico</pais>
        <cp>72320</cp>
    </cliente>
    <orden num='4'>
        <cancion id='14'>
            <nombre>Mi credo</nombre>
            <artista>K-Paz de la Sierra</artista>
            <duracion>00:03:09</duracion>
            <tamano>7.22</tamano>
```

```
            <precio>9.45</precio>
        </cancion>
        <cancion id='11'>
            <nombre>No me digas que no</nombre>
            <artista>Nikki Clan</artista>
            <duracion>00:02:56</duracion>
            <tamano>4.05</tamano>
            <precio>9.25</precio>
        </cancion>
        <cancion id='5'>
            <nombre>Somos novios</nombre>
            <artista>Luis Miguel</artista>
            <duracion>00:03:10</duracion>
            <tamano>2.9</tamano>
            <precio>10.45</precio>
        </cancion>
        <numCanc>3</numCanc>
        <subtotal>29.15</subtotal>
        <iva>4.37</iva>
        <total>33.52</total>
    </orden>
</compra>
```

Figura 4.5: Archivo XML para la compra, compra.xml

Ya que se tiene el archivo XML creado utilizando un servlet se prosigue a la elaboración de su schema y hoja de estilos que se observan en las Figuras 4.6 y 4.7. El schema describe la sintaxis del documento XML relativo a la compra con el fin de definir dicho documento. Por otro lado, a través de la hoja de estilos es posible mostrar el contenido estructurado del documento XML de la compra con un formato determinado, en este caso el HTML.

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
    <xs:element name='compra'>
        <xs:complexType>
            <xs:sequence>
                <xs:element name='fecha' type='xs:date'/>
                <xs:element name='hora' type='xs:time'/>
                <xs:element name='cliente'>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name='nombre' type='string'/>
                            <xs:element name='apellidos' type='string'/>
                            <xs:element name='direccion' type='string'/>
                            <xs:element name='ciudad' type='string'/>
                            <xs:element name='estado' type='string'/>
                            <xs:element name='pais' type='string'/>
                            <xs:element name='cp' type='integer'/>
                        </xs:sequence>
                        <xs:attribute name='email' type='xs:string'/>
                    </xs:complexType>
                </xs:element>
```

```
            <xs:element name='orden'>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name='cancion' maxOccurs='unbounded'>
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name='nombre' type='string'/>
                                    <xs:element name='artista' type='string'/>
                                    <xs:element name='duracion' type='time'/>
                                    <xs:element name='tamano' type='decimal'/>
                                    <xs:element name='precio' type='decimal'/>
                                </xs:sequence>
                                <xs:attribute name='id' type="xs:integer"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name='numCanc' type='integer'/>
                        <xs:element name='subtotal' type='decimal'/>
                        <xs:element name='iva' type='decimal'/>
                        <xs:element name='total' type='decimal'/>
                    </xs:sequence>
                    <xs:attribute name='num' type="xs:integer"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Figura 4.6: Schema para la compra, compra.xsd

```
<xsl:stylesheet mlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="html"/>

    <xsl:template match="/">
        <HTML>
            <BODY text='#28166F' link='#0093dd' topMargin='0'>
                <xsl:apply-templates/>
            </BODY>
        </HTML>
    </xsl:template>

    <xsl:template match="compra">
        <h2 align='center'>Compra no. <xsl:value-of select='orden/@num'/></h2>
        Fecha: <xsl:value-of select='fecha'/>
        Hora: <xsl:value-of select='hora'/>
        <xsl:apply-templates select='cliente'/>
        <xsl:apply-templates select='orden'/>
    </xsl:template>

    <xsl:template match="cliente">
        <H3 align='center'>Datos del comprador:</H3>
        <TABLE border='1' width='80%' bordercolor='#0093DD' align='center'>
            <TR bgcolor='#dff4ff'><TD>
                <xsl:value-of select='nombre'/>
                <xsl:value-of select='apellidos'/>. <br/>
                <xsl:value-of select='direccion'/>. <br/>
```

```
                <xsl:value-of select='ciudad'/>,
                <xsl:value-of select='estado'/>,
                <xsl:value-of select='pais'/>. <br/>
                C.P. <xsl:value-of select='cp'/>.
             </TD></TR>
        </TABLE>
    </xsl:template>

    <xsl:template match="orden">
        <H3 align='center'>Detalle de la orden:</H3>
        <TABLE ALIGN='center' width='100%' CELLSPACING='1' CELLPADDING='1'>
            <TR>
                <TD align='center'> Canciones: <xsl:value-of select='numCanc'/> </TD>
            </TR>
            <TR bgcolor='#0093dd'>
                <TH>Titulo de la cancion</TH>
                <TH>Artista</TH>
                <TH>Duracion</TH>
                <TH>Tamano</TH>
                <TH>Precio</TH>
            </TR>
            <xsl:for-each select='.'>
                <xsl:apply-templates select='cancion'/>
            </xsl:for-each>
        </TABLE>
        Subtotal: $<xsl:value-of select='subtotal'/><BR/>
        IVA: $<xsl:value-of select='iva'/><BR/>
        Total: $<xsl:value-of select='total'/>
    </xsl:template>

    <xsl:template match='cancion'>
        <TR bgcolor='#dff4ff'>
            <TD> <xsl:value-of select='nombre'/> </TD>
            <TD> <xsl:value-of select='artista'/> </TD>
            <TD align='center'> <xsl:value-of select='duracion'/> </TD>
            <TD align='center'> <xsl:value-of select='tamano'/> MB </TD>
            <TD align='center'> $<xsl:value-of select='precio'/> </TD>
        </TR>
    </xsl:template>
</xsl:stylesheet>
```

Figura 4.7 Hoja de estilos para la compra, compra.xsl

Ahora que ya se cuenta con el archivo XML con sus correspondiente Schema y hoja de estilos se prosigue a transformar el archivo XML en el formato HTML. Para ello se utiliza el método que se muestra en la Figura 4.8. Se utilizo la clase Transformer de javax.xml.transform. Una instancia de esta clase puede ser obtenida con el metodo TransformerFactory.newTransformer. Esta instancia puede entonces ser usada para procesar XML de una variedad de fuentes y escribir la salida de la transformación a una variedad de formatos, en este caso procesar el archivo XML de la compra en el formato HTML.

```
/* Transformar un archivo XML en HTML */
public void transformar(String archXSL,String archXML,String archHTML)
    throws TransformerException,TransformerConfigurationException,
    FileNotFoundException,IOException
{
    private TransformerFactory tFactory = TransformerFactory.newInstance();
    private Transformer transformer= tFactory.newTransformer (new StreamSource(archXSL));
    transformer.transform (new StreamSource(archXML), new StreamResult (new
FileOutputStream(archHTML)));
}
```

Figura 4.8: Código del método empleado para transformar un archivo XML a un HTML.

El resultado de la transformación del archivo XML llamado compra.xml al formato HTML da como resultado el archivo compra.html que se puede ver en la Figura 4.9. Con el cual se concluye el cometido, ahora se puede señalar que al cubrir dicha tarea se obtienen los siguientes beneficios y resultados:

El diseño de los tipos documentos necesarios, usando XML, para las compras de la aplicación. Así como ofrecer mecanismos más versátiles de mostrar datos. A través de HTML, representa documentos XML en los navegadores Web. Por otro lado, se efectuó un proceso de datos tradicionales, donde XML codifica los datos para que los procese un programa, en particular Java, donde los documentos XML son contenedores que construyen interfaces. Finalmente, la información de los documentos XML es mas rica y fácil de usar, por las habilidades hipertextuales de XML. Además, la información será mas accesible y reutilizable, por la flexibilidad de las etiquetas de XML y pueden utilizarse sin tener que amoldarse a las reglas especificas de un fabricante. De esta manera la información puede ser usada de diferentes formas por diferentes aplicaciones. Por ejemplo en las búsquedas, ya que los documentos están etiquetados es posible localizar la información de forma más clara.

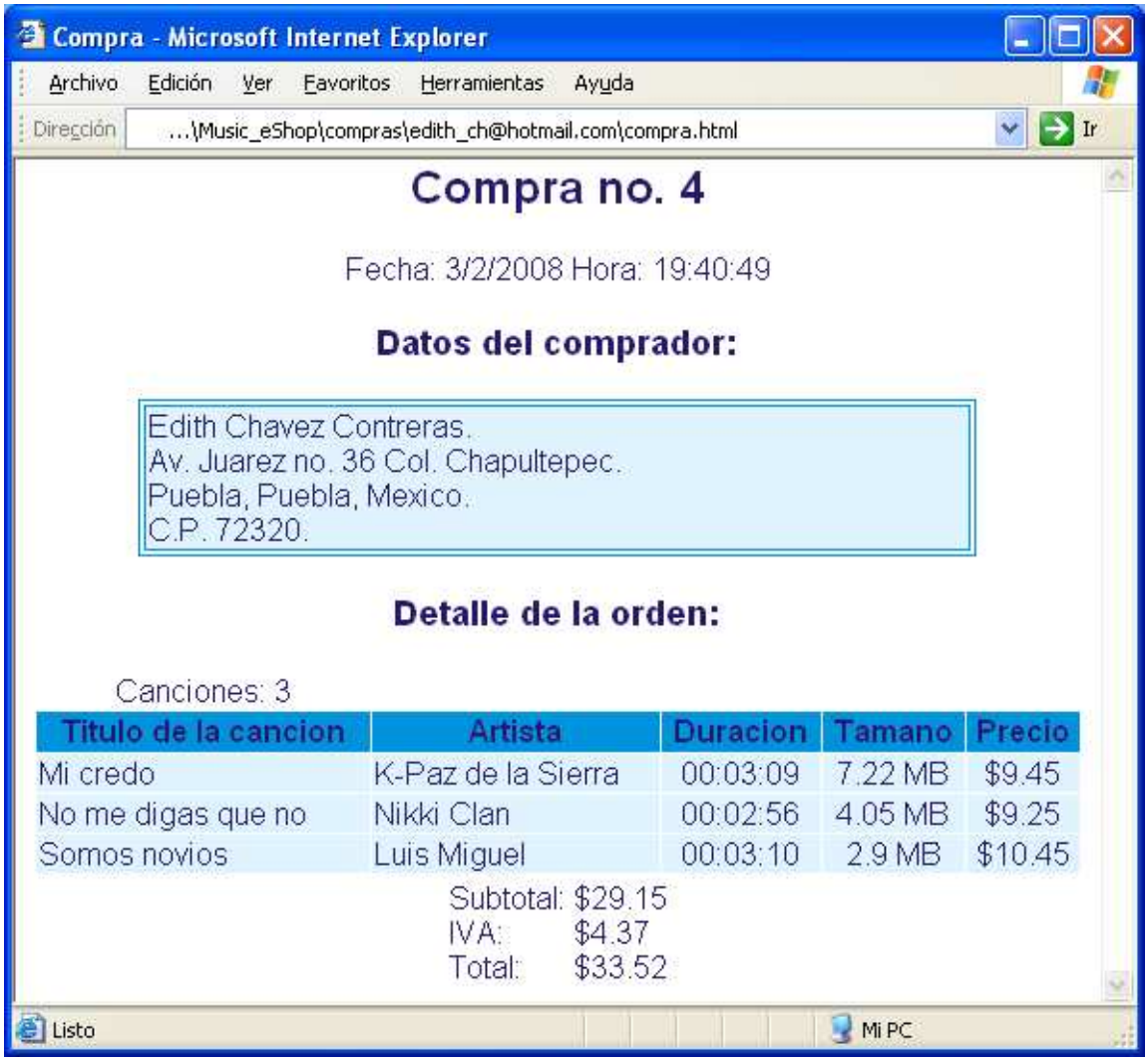

Figura 4.9: Archivo HTML para la compra, compra.html.

4.3 Aplicación Music e-Shop en Acción

En esta sección se presenta el resultado de la implementación de la aplicación Web Music e-Shop. La cual sigue el modelado desarrollado en los dos capítulos anteriores, el análisis y diseño de la tienda on-line utilizando UWE. Para implementar la aplicación Music e-Shop se desarrollaron servlets Java y una base de datos, además de la implantación de XML. La cual se mensiona en el apartado anterior.

En la Figura 4.10 se observa la representación de la clase de navegación principal de la tienda on-line: Home Music e-Shop. En la figura se puede apreciar el formulario de búsqueda, el índice que describe el catálogo y demás elementos descritos en la clase

de navegación Home Music e-Shop. En este caso de ejemplo se asume que el Cliente selecciona al género musical Duranguense.

Figura 4.10: Página Principal, Home Music e-Shop.

A continuación, se despliega la clase de navegación para el detalle del género. En la Figura 4.11 se pueden apreciar los elementos descritos por la clase de navegación Género. Uno de sus elementos es el índice de las Canciones, es decir el listado de canciones correspondientes al género seleccionado. En índice se asume que el Cliente seleccione la canción "Mi credo". Otro elemento es el formulario Agregar Canción a Carrito que contiene al botón correspondiente para esta tarea. Cuando el Cliente presiona este botón, la o las canciones elegidas son enviadas al carrito para continuar con el proceso de compra.

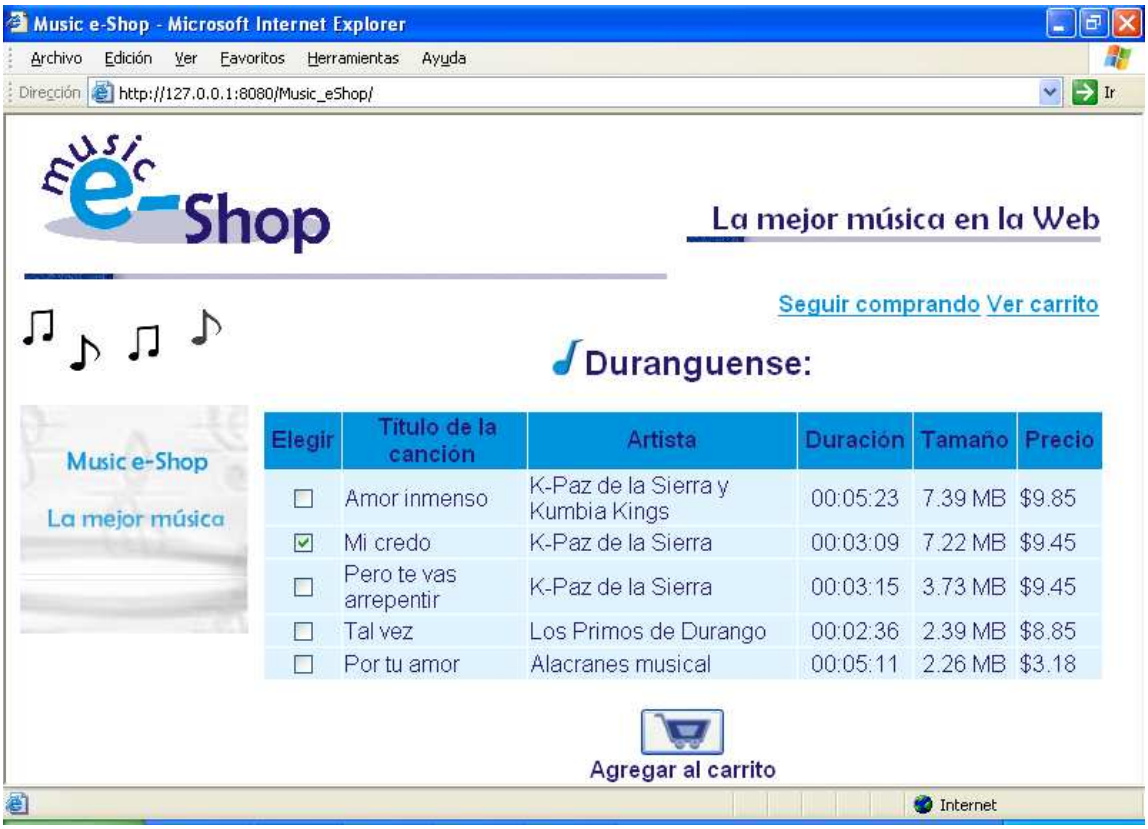

Figura 4.11: Página de un Género Musical (Duranguense).

En la Figura 4.12 se presenta la clase de navegación Carrito. Se observa la lista de artículos, el monto total y las opciones de actualizar carrito para eliminar una canción, seguir comprando u ordenar para concretar la compra. Se puede destacar la manera en como se implementa el listado de canciones. A cada elemento de este listado y al carrito se asocian dos formularios que en conjunto cuentan con los campos requeridos para llevar a cabo la actualización del carro; es decir para sacar artículos del carro si el Cliente lo desea. Con el propósito de visualizar el listado de artículos del carrito, para el caso de ejemplo, se asume que además de la canción “Mi credo” el Cliente también agregó al carrito las canciones “No me digas que no” y “Somos Novios”. Cuando el Cliente elige pasar a pagar se carga la clase de navegación Login cliente que se muestra en la Figura 4.13.

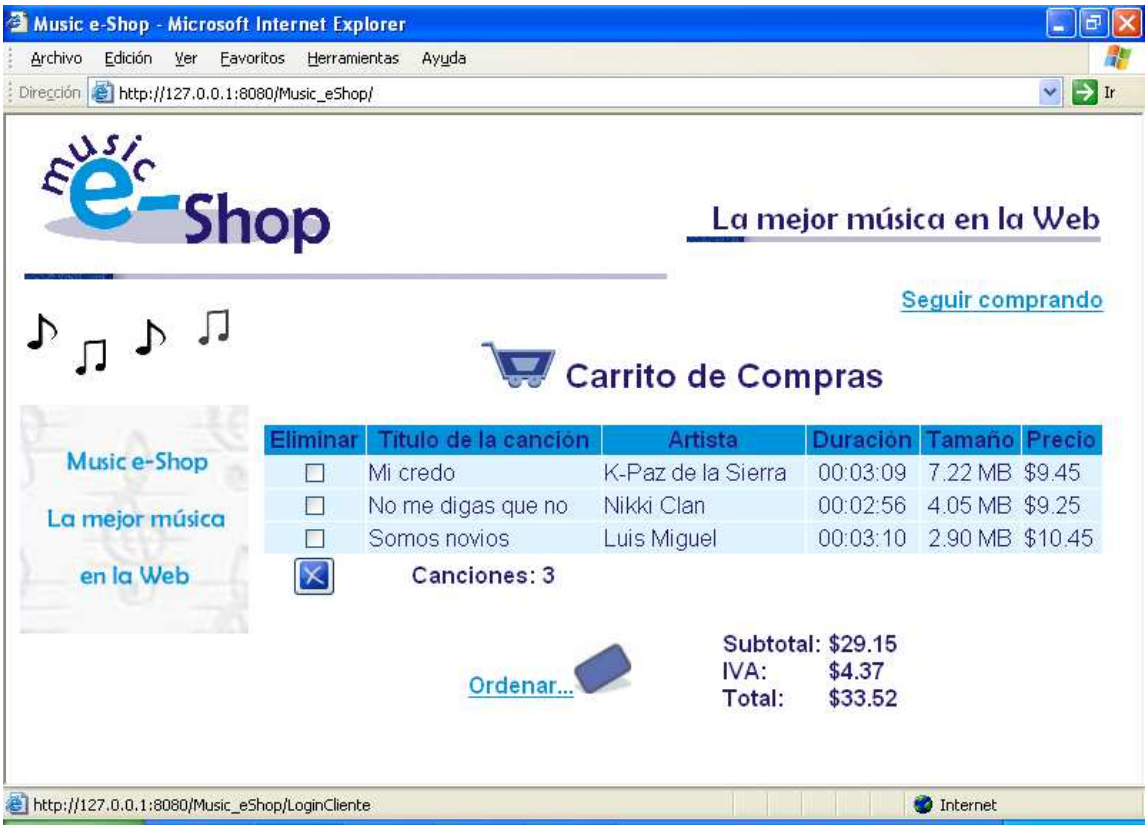

Figura 4.12: Página del Carrito de Compras.

En la Figura 4.13 se presenta la clase de navegación para el logeo del Cliente. En el caso ideal el Cliente ya esta registrado por lo que introduce sus datos de acceso al formulario y presiona el botón Iniciar sesión. Los datos son enviados a la clase de navegación Autenticar Usuario que es transparente al usuario, es decir, que dicha clase de navegación realiza el procesamiento de verificación de los datos de acceso del Cliente y, de acuerdo al resultado de dicho proceso, redirecciona de manera automática hacia la clase de navegación correspondiente sin estacionarse en el navegador para que el Cliente interactué con ella. Este proceso se puede apreciar en la secuencia que describe la autenticación de un cliente existente. En el caso de ejemplo vamos a suponer que el Cliente se autentica correctamente y, de esta manera, el proceso de pago se inicia correctamente en la clase de navegación Información de Pago que se presenta en la Figura 4.14.

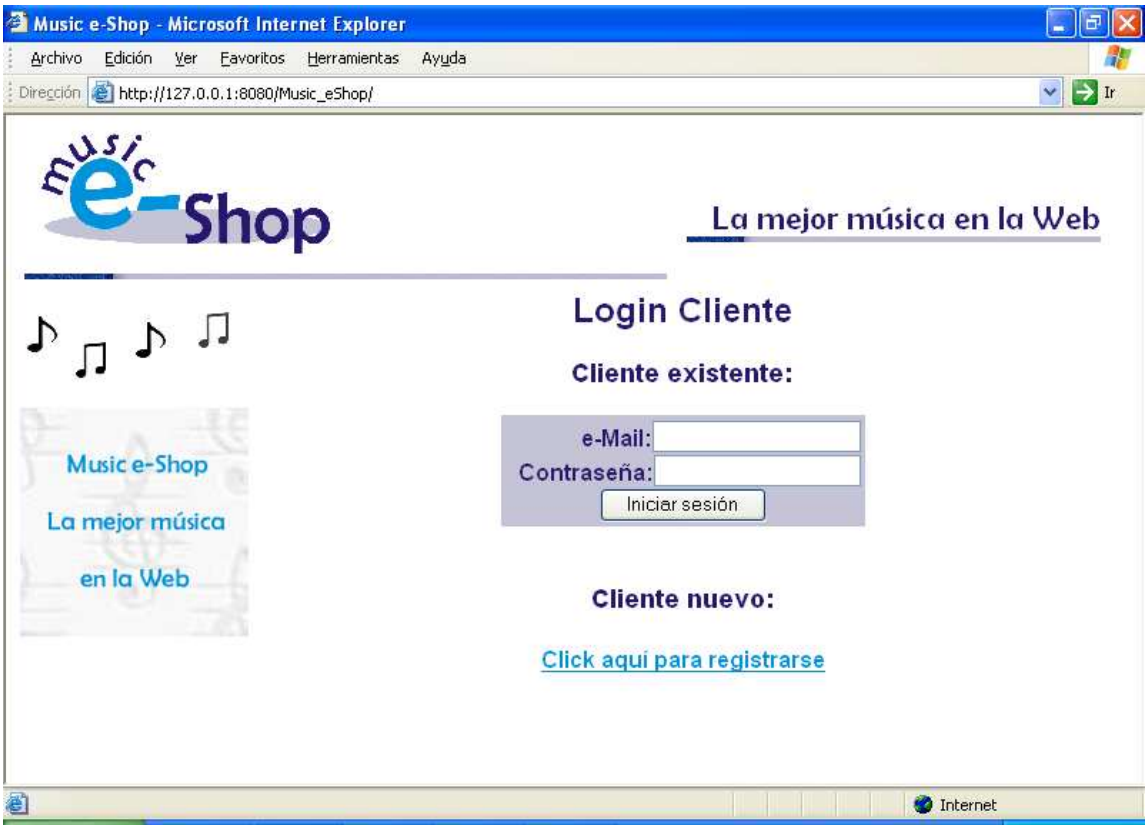

Figura 4.13 Página del Login Cliente.

En la Figura 4.14 se presenta la clase de navegación Información de Pago. En esta clase de navegación el Cliente debe realizar el pago de las canciones a través de su tarjeta bancaria. Antes, el Cliente tiene la posibilidad de modificar los datos personales de su registro. En esta clase de navegación se vuelven a desplegar los datos de la orden y el detalle de las canciones incluidas en ella. Por último, se destaca el formulario mediante el cual el Cliente introduce la información de la tarjeta bancaria con la cual desea realizar el pago de la orden. Una vez que el Cliente ha llenado los campos requeridos, procede a intentar el pago.

Figura 4.14: Página de la Información de Pago.

La información de pago, de acuerdo al modelo, es enviada a la clase de navegación Proceso Pago Seguro. Al igual que autenticación usuario, Proceso pago seguro es transparente al usuario, es decir, no se estaciona en el navegador para que ningún usuario pueda interactuar con él. Cabe señalar que en el presente proyecto solo se simulo el proceso de pago seguro ya que no se maneja la interacción con la entidad bancaria para efectuar una transacción. Como resultado del proceso, si el pago se llevó a cabo de manera exitosa, el Cliente únicamente visualiza en su navegador la

confirmación del pago. Este es el caso que ocurre en el caso de ejemplo y que se muestra en la Figura 4.15. Si el pago no es exitoso, la notificación para el Cliente será de un error.

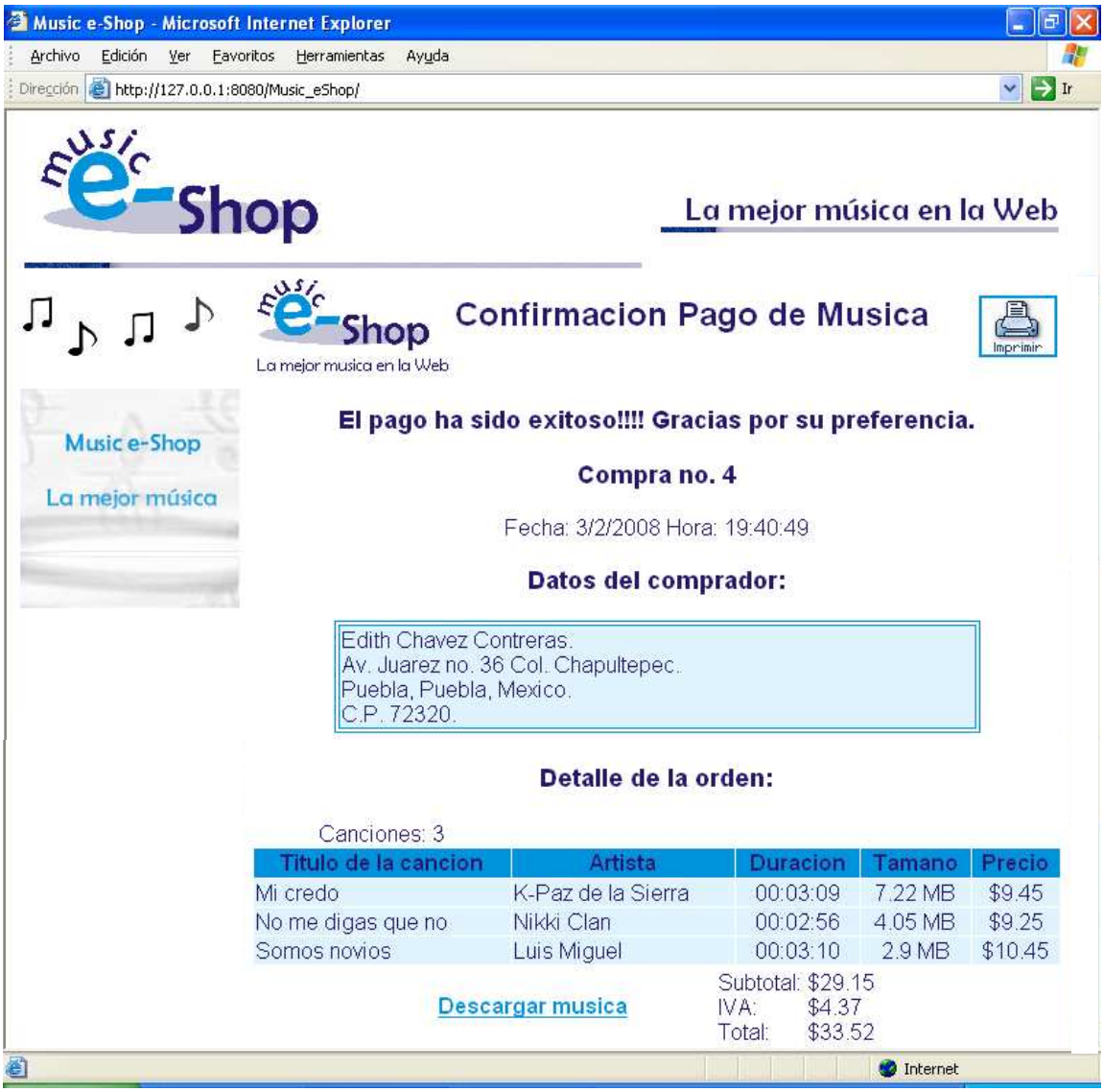

Figura 4.15: Pagina Confirmación de Pago de Música (exitoso).

Todas las funcionalidades de Music e-Shop, al igual que el caso de ejemplo expuesto en este apartado, se han modelado e implementado bajo el mismo patrón de lógica y metodología. En consecuencia, se puede afirmar que el modelo en UML de Music e-Shop y su implementación desarrollados en esta tesis, revisado con el la atención pertinente, constituyen una referencia válida y útil como un prototipo de implementación, así como también, un instrumento de documentación y verificación de la aplicación.

Conclusiones

Para finalizar se presentan las conclusiones obtenidas del desarrollo de esta tesis, cuyo objetivo principal desde un inicio fue el de diseñar una aplicación de comercio electrónico, aplicando UWE como una metodología apropiada para el modelado de aplicaciones Web. Construir la aplicación siguiendo su modelado, así como permitir guardar en un archivo XML los datos extraídos de una base de datos. También, permitir leer dichos archivos XML y analizarlos, así como mostrar los datos contenidos en aquellos archivos en formato HTML. Para así, de esta manera proporcionar recursos tecnológicos (específicamente de software) para la realización de comercio electrónico.

En torno a este objetivo se concluye lo siguiente:

- Se realizó el análisis, diseño e implementación de una Tienda Virtual. Resultado de este trabajo se obtuvo: la especificación de requisitos, modelo lógico-conceptual, modelo de navegación, modelo de interacción temporal y colección de escenarios Web, utilizando UWE. Se generó la implementación de una aplicación utilizando una base de datos relacional con intercambio de información con la aplicación Web bajo el estándar XML y el manejo de parsers para el análisis de documentos XML y transformaciones a otros formatos como el HTML.
- En la realización de cada uno de los diagramas del modelo de Music e-Shop aquí presentado se utilizó el Lenguaje Unificado de Modelado (UML) en el marco de la Ingeniería Web Basada en UML (UWE). UWE es una propuesta enfocada al modelado de aplicaciones Web basada en la extensión de la semántica del UML mediante la utilización de estereotipos.
- Se ejemplifico el uso que se le puede dar a XML para manipular datos extraídos de un almacén de datos y generar otro documento que no dependa del uso de bases de datos.
- El proyecto en su conjunto constituye una fuente de información apropiada para el desarrollo de temas de investigación afines. Además proporciona recursos tecnológicos (específicamente de software) para la realización de Comercio Electrónico.

- Este trabajo constituye una referencia para todos aquellos desarrolladores de aplicaciones Web que deseen incluir, en su proceso de desarrollo, modelos en UML en la constante búsqueda de hacer cada día mejores productos.
- El estándar XML, el gestor de bases de datos MySQL y la arquitectura de software Java que se utilizan en este proyecto son abiertos y no requieren de recursos económicos altos para poder utilizarlos, lo cual propicia más oportunidades de desarrollo para el comercio electrónico.

Trabajo Futuro:

El trabajo futuro inmediato es llevar a cabo el proceso completo de implementación de Music e-Shop propuesto, esto es, concluir la administración de la aplicación, así como, la construcción e incorporación de la entidad bancaria del sistema y la realización de pruebas sobre el mismo. Otros trabajos interesantes a considerarse tendrían que ver con mejoras al prototipo de Music e-Shop aquí presentado. Entre estas mejoras se enumerar las siguientes:

- Modelar e implantar en Music e-Shop la integración de otras opciones de pago distintas al pago con tarjeta de crédito, las cuales puedan coexistir y ser configurables por el Administrador. Por ejemplo opciones de prepago y por transferencia bancaria.
- Modelar en la tienda on-line mecanismos de comunicación directa entre Clientes de Music e-Shop y el Administrador. Por ejemplo foros de opinión, chat y servicios de mensajería.
- En cuanto a lo que se refiere al almacenamiento de información se podrá utilizar lo hecho en XML para reciclar y aprovechar este recurso.

Finalmente, sobre la misma línea de interés de aplicaciones Web de comercio electrónico así como el uso de XML, se puede tomar en cuenta el trabajo realizado hasta este momento como referencia para el desarrollo de proyectos afines.

Apéndice A

Diseño de la Base de Datos de Music e-Shop

Como parte de la Implantación se diseña y construye la base de datos usando el modelo Entidad-Relación y MySQL respectivamente. En el diseño Entidad-Relación de la base de datos, que se muestra en la Figura 36, se crean 10 entidades con sus respectivos atributos; las entidades representan a las tablas de la base de datos y los atributos a sus filas. A continuación se describe brevemente a las entidades:

CLIENTE: Almacena los datos personales de un usuario Cliente de Music e-Shop.

ADMON: Almacena el password de un usuario Administrador de Music e-Shop.

GENERO: Almacena la información correspondiente a un Género Musical del catálogo de la tienda on-line.

CANCION: Almacena la información correspondiente a una canción registrada en el catálogo.

ITEM_GENERO: Representa a una canción que pertenece a un género musical.

ORDEN: Almacena la información correspondiente a una orden de compra efectuada por un Cliente.

ITEM_ORDEN: Representa a una canción como un articulo que pertenece a una orden de compra.

TARJETA_CREDITO: Almacena la información correspondiente a una tarjeta de crédito necesaria para efectuar el pago de una orden.

TARJETA_TIPO: Representa el tipo de de una o mas tarjeta crédito.

Las entidades descritas anteriormente se relacionan de la siguiente manera:un GENERO puede tener uno o más ITEM_GENERO, un ITEM_GENERO debe ser una CANCION. Una CANCION puede ser un ITEM_GENERO y/o una o más CANCIONES pueden pertenecer a una ORDEN. Una ORDEN debe de tener uno o más ITEM_ORDEN, pertenecer a un CLIENTE y ser pagada por una TARJETA_CREDITO que debe de ser de una TARJETA_TIPO. Y una TARJETA_TIPO puede representar a una o más TARJETA_CREDITO. Además un CLIENTE puede efectuar una o más ORDEN y una TARJETA CREDITO puede concretar el pago de una o más ORDEN.

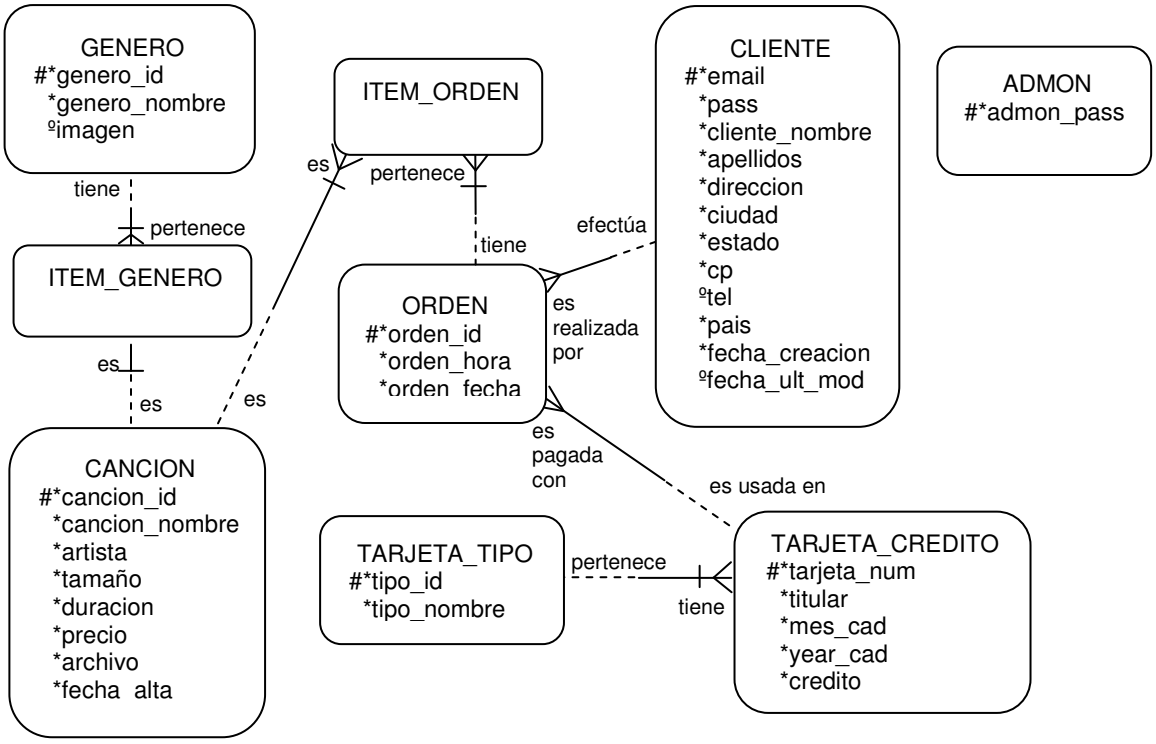

Figura 36 Diagrama Entidad-Relación de la base de datos de Music e-Shop

Una forma de comprobar el contenido de este diagrama entidad-relación es seguir la normalización. La cual es un estándar que consiste, básicamente, en un proceso de conversión de las relaciones entre las entidades, evitando redundancia e inconsistencia de los datos. El proceso de normalización esta conformado por formas normales, de las cuales las fundamentales, aplicadas en este trabajo, son las siguientes.

1ra. forma normal. Eliminar grupos repetidos. En la entidad cliente se dividió el atributo apellidos en apellido_p y apellido_m.

Forma normal 2. Eliminar parte de las dependencias de las llaves. Ya no aplica más ya que todos los atributos de cada entidad dependen directamente de su correspondiente llave primaria.

Forma normal 3. Eliminar dependencia transitiva de los datos. No existen en este caso.

Forma Normal de Boyce-Codd (BCNF). Eliminar las dependencias de una llave intermedia. No existe en este caso.

Ahora se representa la base de datos por medio de tablas, éstas obtenidas mediante el proceso de normalización. Dichas tablas se construyen utilizando MySQL y siguiendo la delineación del modelo relacional.

Tablas de la base de datos

Nombre de tabla: GENERO

Nombre columna	Tipo llave	Nulos	Tipo
genero_id	PK	NN	Integer
genero_nombre		NN	String
imagen			String

Nombre de tabla: CANCION

Nombre columna	Tipo llave	Nulos	Tipo
cancion_ id	PK	NN	Integer
cancion_ nombre		NN	String
artista		NN	String
tamaño		NN	Decimal
duracion		NN	Decimal
precio		NN	Money
archivo		NN	String
fecha_alta		NN	Date
fecha_baja			Date

Nombre de tabla: ITEM_GENERO

Nombre columna	Tipo llave	Nulos	Tipo
genero	FK, PK	NN	Integer
canción	FK, PK, UK	NN	Integer

Nombre de tabla: CLIENTE

Nombre columna	Tipo llave	Nulos	Tipo
email	PK	NN	String
pass		NN	String
cliente_ nombre		NN	String
apellido_p		NN	String
apellido_m		NN	String
direccion		NN	String
ciudad		NN	String
estado		NN	String
cp		NN	Decimal
tel			Decimal
pais		NN	String
fecha_ creacion		NN	Date
fecha_ ult_mod			Date

Nombre de tabla: TARJETA_TIPO

Nombre columna	Tipo llave	Nulos	Tipo
tipo_id	PK	NN	Integer
tipo_nombre		NN	String

Nombre de tabla: TARJETA_CREDITO

Nombre columna	Tipo llave	Nulos	Tipo
tarjeta_tipo	FK, PK	NN	Integer
tarjeta_num	PK	NN	Decimal
titular		NN	String
mes_cad		NN	Decimal
year_cad		NN	Year
credito			Money

Nombre de tabla: ORDEN

Nombre columna	Tipo llave	Nulos	Tipo
orden_id	PK	NN	Integer
orden_fecha		NN	Date
cliente	FK	NN	String
tarjeta_tipo	FK	NN	Integer
tarjeta_num	FK	NN	Decimal

Nombre de tabla: ITEM_ORDEN

Nombre columna	Tipo llave	Nulos	Tipo
orden	FK, PK	NN	Integer
cancion	FK, UK	NN	Integer

Nombre de tabla: ADMON

Nombre columna	Tipo llave	Nulos	Tipo
admon_pass	PK	NN	String

Apéndice B

Creación y Configuración de la Plataforma de Desarrollo de Aplicaciones Web con Java y MySQL

En el capítulo 4 se mencionaron las herramientas de desarrollo empleadas en el sistema. Complementariamente, en este apéndice se comenta brevemente la creación y configuración de la plataforma de desarrollo de aplicaciones Web con Java y MySQL utilizada en el desarrollo de la aplicación Music e-Shop. Para ello se necesita instalar y configurar lo siguiente:

J2SE

Java 2 Plataform Standard Edition Development Kit (abreviadamente, J2SE Developmet Kit o JDK) proporciona la base para desarrollar y distribuir aplicaciones que se podrán ejecutar en un servidor o en un ordenador personal con distintos sistemas operativos. Actualiza las versiones anteriores e incluye nuevas características, pero conservando la compatibilidad y la estabilidad.

Para instalar el paquete de desarrollo, en este caso J2SE 5.0, se descarga el J2SE de la página java.sun.com/j2se, se ejecuta y se siguen los pasos indicados durante la instalación hasta finalizarla.

Gestor de bases de datos MySQL

MySQL, como se menciono en el capítulo 4, es un gestor de bases de datos relacionales. El cual es un software de libre distribución, por lo que se puede obtener una versión actualizada de la dirección de Internet http://dev.mysql.com. En este caso se utilizo la versión 5.0.

Para instalar MySQL primero se descomprime el fichero mysql-5.0.45-win32.zip, en seguida, se ejecuta el fichero setup.exe y se siguen los pasos indicados durante la instalación. Cuando finalice se ejecuta el asistente de configuración. Éste entre otras cosas, permite establecer la contraseña para el usuario root.

Una vez realizado lo anterior, se arranca el monitor MySQL, con el que se puede realizar cualquier operación permitida sobre una base de datos, ejecutando la sentencia SQL apropiada.

Controlador MySQL

Después de la instalación de MySQL, es necesario instalar también el controlador para poder conectarse a MySQL vía JDBC (Java DataBase Conectivity). Para ello, se descarga una versión reciente de la dirección de Internet www.mysql.com/dowloads/connector. Después, se descomprime el fichero .zip que fue descargado y se copia el fichero mysql-connector-java-x.x.x-bin.jar en la carpeta …/jre/lib/ext de la instalación de java.

Tomcat

Tomcat es un es un servidor de aplicaciones que implementa las tecnologías Java Servlet y JavaServer Pages. Se puede obtener de la dirección de Internet. jakarta.apache.org/tomcat. Tomcat además de ser un contenedor Web, puede trabajar como un servidor Web por lo que permite servir tanto paginas estáticas como dinámicas.

Suponiendo que ya se tiene instalado J2SE, la instalación sobre Windows de Tomcat 5.0 se realiza de la siguiente manera: se ejecuta el fichero .exe previamente descargado de Internet. El instalador utilizará la variable de entorno JAVA_HOME registrada en el registro de Windows para determinar la ruta de la maquna virtual de Java en el JDK o en el JRE. También creara accesos directos que permitirán arrancar y configurar el servidor. Se siguen los pasos indicados durante la instalación.

Para comprobar que la instalación ha sido exitosa se arranca el servidor y se introduce la dirección http://localhost:8080/ en el navegador, y se debería ver la página de bienvenida de Tomcat.

Una vez que ha sido instalado correctamente, se edita el fichero <TOMCAT_HOME>/conf/web.xml y se permite que, durante la fase de desarrollo, se ejecute el código siguiente:

```
<servlet>
<servlet-name>invoker</servlet-name>
<servlet-class>
  org.apache.catalina.servlets.InvokerServlet
< /servlet-class>
<init-param>
  <param-name>debug</param-name>
  <param-value>0</param-value>
</init-param>
<load-on-startup>
</servlet>
…
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
<servlet-mapping>
```

Este código, por seguridad viene comentado, Si durante la fase de desarrollo se permite que se ejecute, se podrá invocar la ejecución de servlets anónimos de la forma: http://servidor[:puerto]/localización/servlet/nombre_servlet.

Para dar acceso a Tomcat (y sus aplicaciones) a base de datos, se debe copiar el fichero del controlador JDBC de la base de datos al directorio common/lib de Tomcat. Finalmente, se debe configurar los roles admin y manager de Tomcat, y los nombres de usuarios y las passwords. Se edita el fichero tomcat-users.xml en el directorio conf de Tomcat, para que se parezca a esto:

```
<?xml version='1.0'?>
   <tomcat-users>
      <role rolename="admin"/>
      <role rolename="manager"/>
      <user username="tomcatusername" password="tomcatpassword"
            roles="admin,manager"/>
   </tomcat-users>
```

Es importante notar que el administrador de aplicaciones de Tomcat solo podrá ser utilizado cuando el servidor esté arrancado.

Después de instalar y configurar J2SE, MySQL, el controlador MySQL y Tomcat ya se cuenta con lo necesario para desarrollar aplicaciones Web con Java y MySQL; específicamente, en el caso de este proyecto empleando servlets Java.

Glosario

API es el acrónimo de Interfaz de Programación de Aplicaciones (Application Programming Interface).

ASCII, Código Normalizado Americano para el Intercambio de Información (American Standard Code for Information Interchange), es un esquema estándar de codificación de caracteres utilizado por la mayoría de los ordenadores para visualizar letras, dígitos y caracteres especiales.

CSS, Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo para añadir estilo a documentos HTML y XML.

DOM, Modelo de Objetos del Documento (Document Objet Model), es una API que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, navegar por su estructura y, modificar, añadir y borrar tanto elementos como contenidos.

DTD, Definición de Tipo de Datos (Document Type Definition), es una colección de reglas usadas con el propósito de identificar un tipo o clase de documento.

EBNF es el acrónimo de la Forma Extendida de Backus Naur (Extended Backus Naur Form). EBNF es una notación para describir formalmente la sintaxis de un lenguaje de programación.

FTP, Protocolo de Transferencia de Archivos (File Transfer Protocol), es un protocolo de transferencia de archivos que se utiliza en Internet y otras redes para transmitir archivos entre servidores o entre un usuario y un servidor.

HTML, Lenguaje de Etiquetado de Hipertexto (HiperText Markup Language), es un lenguaje de marcas comúnmente utilizado para la publicación de hipertexto en la Web.

HTTP, Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol), es un protocolo utilizado para la transferencia de datos a través de Internet, y que está basado en operaciones sencillas de solicitud y respuesta.

IP es el acrónimo de Protocolo de Internet (Internet Protocol).

IRC, Canal de Chat de Internet (Internet Realy Chat), es un servicio de Internet para la comunicación en tiempo real entre usuarios.

JDBC, acrónimo de Conexión con Bases de Datos desde Java (Java DataBase Conectivity).

SAX, API simple para XML(Simple API for XML), es un analizador sintáctico secuencial.

SGML, Lenguaje Estándar de Etiquetado Generalizado (Estandar Generalized Markup Language), es un estándar internacional para el intercambio y almacenamiento de información.

TCP es el acrónimo del Protocolo de Control de Transmisión (Transmisión Control Protocol).

TCP/IP (Transmisión Control Protocol/ Internet Protocol) es un conjunto de protocolos usados en Internet.

Telnet (Tele Network) es un servicio de Internet que permite acceder remotamente a otro ordenador de la Red.

UML es el acrónimo del Lenguaje Unificado de Modelado (Unified Model Language).

UWE es el acrónimo de la Ingeniería Web basada en UML (UML-based Web Engineering).

WWW es el acrónimo del "World Wide Web". Es un sistema hipermedia interactivo que permite conectarse a grandes cantidades de información en Internet.

XML, Lenguaje de Etiquetado Extensible (eXtensible Markup Language), es un lenguaje con una importante función en el proceso de intercambio, estructuración y envío de datos en la Web.

XML Schema, Esquema XML, es un lenguaje cuyo objetivo principal es definir la estructura en bloques de un documento XML.

XSL, Lenguaje de Hojas de Estilo Extensible (eXtensible Style sheet Language), Es un lenguaje para crear hojas de estilo a través de las cuales será posible mostrar el contenido estructurado de un documento con un formato determinado.

XSLT, Transformaciones del Lenguaje de Hojas de Estilo Extensible (eXtensible Style sheet Language Transformations), es un lenguaje que permite la transformación de la estructura de un documento XML en otro documento XML con estructura diferente.

Bibliografía

[Mateu04] MATEU Carles, “Desarrollo de Aplicaciones Web”, Formación de Postgrado, Marzo 2004.

[W3C] W3C, Página del World Wide Web Consortium: www.w3c.org

[Charles99] CHARLES F. Goldfarb & Paul Prescod. "Manual de XML." Ed. Prentice Hall. 1999.

[Vázquez02] VAZQUEZ Adolfo, “Navegar en Internet, XML” Ed. Alfaomega Ra-Ma, 2002.

[Rossainz06] ROSSAINZ Mario, “Desarrollo de Aplicaciones Web”, Diplomado en Computación: Tecnologías de la información, Benemérita Universidad Autónoma de Puebla, 2006.

[Gutiérrez01] GUTIÉRREZ Abraham y Martínez Raúl, “XML a través de ejemplos”, Ed. Alfaomega Ra-Ma, 2001.

[Ocaña06] OCAÑA Jesús, “Análisis y Diseño en UML de un Centro Comercial Virtual enfocado a Pequeñas y Medianas Empresas(PyMEs) Mexicanas”, Benemérita Universidad Autónoma de Puebla, maestría en Ciencias, Disertación, Septiembre de 2006

[RossOcaUWE06] ROSSAINZ Mario, OCAÑA Jesús, “Introducción a la Ingeniería Web Basada en UML”, Benemérita Universidad Autónoma de Puebla, Ciencias de la Computación, 2006.

[RossOcaCCV06] ROSSAINZ Mario, OCAÑA Jesús, “Modelado de un Centro Comercial Virtual usando la Ingeniería Web basada en UML para Pequeñas y Medianas Empresas (PyMEs) Mexicanas”, Benemérita Universidad Autónoma de Puebla, Ciencias de la Computación, 2006.

[Ceballos03] CEBALLOS Fco. Javier, JAVA 2 “Interfaces gráficas y aplicaciones para Internet”, Ed. Alfaomega Ra-Ma, 2da. Edición, 2003.