UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
OPCIÓN PROFESIONAL: INGENIERÍA DEL SOFTWARE
E INTERACCIÓN HUMANO-COMPUTADOR

DESARROLLO DE UNA APLICACIÓN WEB DE PROGRAMA DE INCENTIVOS BAJO LA ARQUITECTURA CLIENTE-SERVIDOR UTILIZANDO LA METODOLOGÍA DE DESARROLLO RUP-SCRUM

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE
UNIVERSIDAD CENTRAL DE VENEZUELA
POR LA BACHILLER:

ANDRADE PERAFÁN, VERÓNICA DALILA

PARA OPTAR AL TÍTULO DE
LICENCIADA EN COMPUTACIÓN

TUTOR:

PROF. FERNÁNDEZ, JUAN CARLOS

CARACAS, OCTUBRE 2009

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

ACTA

Quienes suscriben miembros del Jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la bachiller **Verónica D. Andrade P.**, **C.I. 16.473.324** con el título **"Desarrollo de una Aplicación Web de Programa de Incentivos bajo la arquitectura Cliente-Servidor utilizando la metodología de desarrollo RUP-Scrum"**, a los fines de optar al título de Licenciada en Computación, dejan constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijó el 23 de Octubre de 2009 a las 11:00 a.m., para que su autor lo defendiera en forma pública, se hizo en el aula PB III de la Escuela de Computación, mediante una presentación oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el Jurado decidió aprobarlo con una nota de _____ puntos, en fe de lo cual se levanta la presenta Acta, en Caracas a los 23 días del mes de Octubre del año dos mil nueve.

_____
Prof. Juan Carlos Fernández Perregil
Tutor

_____ _____
Prof. Brenda López **Prof. Jossie Zambrano**
Jurado Principal **Jurado Principal**

_____ _____
Prof. Keyla Rivas **Prof. Robinson Rivas**
Jurado Suplente **Jurado Suplente**

RESUMEN

El objetivo de este Trabajo Especial de Grado es construir una aplicación web que sirva como base para el desarrollo de programas de incentivos, a través de la fácil modificación de la interfaz según la imagen de la empresa así como las reglas de negocio que deben aplicar, utilizando una metodología de desarrollo híbrida entre RUP y Scrum, donde se tomen los aspectos de cada una que permitan desarrollar un buen producto con tiempos de espera cortos.

Un programa de incentivos pretende incentivar la fuerza de trabajo de una empresa en la promoción y venta de los productos que ofrece dicha empresa.

El desarrollo de la aplicación web se llevó a cabo utilizando como método de desarrollo una combinación entre RUP y Scrum. El trabajo se dividió en cuatro (4) fases según la metodología RUP (a saber): inicio, elaboración, construcción y transición. En la fase de inicio se analizó el problema, se determinaron las principales funcionalidades del sistema y se realizaron las primeras versiones del modelo conceptual, del modelo de datos y de la interfaz del sistema tomando en cuenta la navegabilidad dentro de la aplicación web. En la fase de elaboración se profundizó el análisis y se determinaron el resto de las funcionalidades del sistema. La fase de construcción se desarrolló de acuerdo a la metodología Scrum realizando un listado de las funcionalidades del sistema agrupadas por módulos, según su afinidad, e implementándolas semanalmente de modo que al final de cada semana se obtuvo un incremento funcional de la aplicación. Finalmente en la fase de transición se instaló el producto y se realizaron pruebas colectivas mediante una matriz de pruebas.

Esta combinación de metodologías de desarrollo de software permitió formalizar la documentación del proyecto, a través de los diferentes diagramas realizados, y minimizar los tiempos de entrega, al dividir el trabajo de tal forma que semanalmente se obtuviera un incremento funcional del sistema. A la vez que se logró un producto de calidad que cumplió con las expectativas del usuario final.

Palabras clave: Programa de Incentivos, Carrito de Compra, Metodología, Lealtad, Multimoneda, Tienda Virtual de Recompensas.

ÍNDICE GENERAL

CAPÍTULO 3: MARCO APLICATIVO

ÍNDICE DE TABLAS Y FIGURAS

FIGURAS

TABLAS

INTRODUCCIÓN

Las empresas siempre han buscado herramientas para mejorar su productividad con el fin último de aumentar sus ganancias. Una de esas herramientas son los programas de incentivos.

El objetivo de los programas de incentivos es crear un sistema de recompensas equitativas para la organización y los trabajadores. De este modo, los programas de incentivos buscan motivar al trabajador con beneficios que premien su esfuerzo.

Los reconocimientos en el campo laboral son incentivos que premian el esfuerzo del trabajador, su antigüedad y dedicación entre otros factores. De manera que éstos consisten en incentivos para estimular ciertos tipos de comportamiento. Los sistemas de reconocimientos y recompensas que se otorgan al personal de una empresa pública o privada permiten que se premie las conductas positivas en los miembros de una organización.

Para la implementación de éstos se toma en cuenta que deben ser adecuados, equitativos, eficientes en costos, seguros y aceptables para los trabajadores. Como deben ser eficientes en costo y seguros se propuso desarrollar una aplicación web de programas de incentivos, ya que ofrece la automatización de los procesos abarcados dentro de un programa de incentivos disminuyendo los costos y las tasas de errores al momento de la realización de auditorías.

El desarrollo de aplicaciones Web ha tomado auge en los últimos años con la masificación de Internet en el mundo, y cada vez son más las tecnologías disponibles orientadas al desarrollo de este tipo de aplicaciones. Cada día son más las empresas e institutos que están migrando sus procesos administrativos así como los servicios que ofrecen a esta red de la información. De hecho, cada día es mayor el número de desarrolladores Web que se preparan alrededor del mundo en respuesta a esta tendencia.

Todas las formas de hacer programas han ido evolucionando hacia las n-capas, convirtiendo a esta forma de modelar en algo lógico y normal que se realiza actualmente. Con la utilización de la Arquitectura de Software en n-capas, se logra desarrollar aplicaciones de manera clara, nítida y transparente, esto es mediante componentes, con la ventaja de poder desarrollar varias capas al mismo tiempo, también se logran aplicaciones robustas debido al encapsulamiento, mantenimiento más sencillo (arreglar directamente en el componente en cuestión), separación adecuada de funciones, adaptación sencilla de los programas a las nuevas necesidades, además de tener mayor escalabilidad y flexibilidad entre otras ventajas.

De este modo, las arquitecturas de n-capas se están posicionando rápidamente como la piedra angular de los desarrollos de aplicaciones empresariales y las compañías están adoptando esta estrategia a una velocidad de vértigo como mecanismo de posicionamiento en la economía emergente que tiene su base en la red.

Para el funcionamiento de una aplicación Web cualquiera, esta debe ser alojada en un Servidor Web (bien sea que esté instalado localmente en una computadora personal, en una computadora o supercomputadora en otro sitio), que estará a la espera de peticiones por los clientes Web para el acceso a alguno de los elementos de la aplicación Web. Por otra parte, dicha aplicación Web por lo general puede conectarse a una base de datos alojada en un servidor de bases de datos (igualmente instalado local o remotamente), o utilizar otras

tecnologías. Por tanto, es imprescindible que estos elementos estén presentes para que una aplicación Web funcione (principalmente el servidor Web). (Villarroel, 2006)

El alcance de la aplicación y el tipo de usuarios a los que estará dirigida son consideraciones tan importantes como las tecnologías elegidas para realizar la implementación. Así como las tecnologías pueden limitar la funcionalidad de la aplicación, decisiones de diseño equivocadas también pueden reducir su capacidad de extensión y reusabilidad. Es por ello que el uso de un método de diseño y de tecnologías que se adapten naturalmente a ésta, son de vital importancia para el desarrollo de aplicaciones complejas.

Hay literalmente cientos de tecnologías a disposición del webmaster de hoy en día y, aprovechándolas adecuadamente, hacen que el sitio sea dinámico, amigable y exitoso.

La controversia respecto a cuál tecnología usar en el lado del servidor se mezcla con el problema de cuál sistema operativo/arquitectura usar. De esta forma, es un elemento más en la eterna batalla entre el mundo Unix y el mundo Microsoft.

El objetivo de este Trabajo Especial de Grado es construir una aplicación web que sirva como base para el desarrollo de programas de incentivos, a través de la fácil modificación de la interfaz según la imagen de la empresa así como las reglas de negocio que deben aplicar, utilizando una metodología de desarrollo híbrida entre RUP y Scrum, donde se tomen los aspectos de cada una que permitan desarrollar un buen producto con tiempos de espera cortos.

La llegada de las empresas a la estructuración de programas o planes de incentivos es reciente como fruto de una toma de conciencia de la responsabilidad social de la empresa e impulsadas por una serie de factores, tales como, actitud del empleado en cuanto a los beneficios, exigencias sindicales, legislación laboral y seguridad social, competencia entre las empresas para mantener o atraer recursos humanos y contrastes salariales generados indirectamente con el mercado. De aquí la principal motivación de la realización de este Trabajo Especial de Grado, pretendiendo ofrecer una buena propuesta de programa de incentivos que no sea costosa para la empresa y pueda ser implementada y puesta en marcha de forma rápida utilizando una metodología de desarrollo híbrida entre RUP y Scrum.

El Proceso Unificado de Rational (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Scrum es una forma de gestionar proyectos de software. No es una metodología de análisis, ni de diseño, como podría ser RUP, es una metodología de gestión del trabajo. Una de las características más importantes es que es muy fácil de explicar y de entender, lo que ayuda mucho a su implantación. Scrum debe todas sus prácticas desde un proceso iterativo e incremental.

Cada uno de los puntos tratados anteriormente se describen con mayor detalle en los siguientes capítulos descritos a continuación:

Capítulo 1: Marco Conceptual. En este capítulo se describen algunos de los conceptos fundamentales referentes a las aplicaciones web, los cuales representan la base de esta

investigación ya que la misma tiene como objetivo el desarrollo de una aplicación Web Cliente-Servidor. Dichos conceptos se refieren a: Arquitectura Cliente-Servidor con sus respectivos componentes, características y modelos, Aplicaciones Web, donde se detallan los requisitos para el desarrollo de este tipo de aplicaciones y las tecnologías que involucra, tanto del lado del cliente como del lado del servidor. Además se describe el sistema manejador de base de datos SQL Server y se enumeran sus principales ventajas.

Capítulo 2: Marco Metodológico. Las metodologías tomadas en cuenta para el desarrollo de este Trabajo Especial de Grado, metodología RUP y metodología Scrum, son descritas en este capítulo, resaltando las actividades realizadas en cada una así como la división del trabajo y la forma en cómo lo abordan.

Capítulo 3: Marco Aplicativo. Dirigido a presentar el trabajo de investigación y el conocimiento obtenido de las necesidades de desarrollar una aplicación Web de programas de incentivos, definiendo para ello el problema, los objetivos, el límites y alcance, la justificación e importancia, la plataforma tecnológica utilizada en el desarrollo del proyecto y, finalmente, la metodología de desarrollo utilizada para construir la aplicación web de programa de incentivos.

Finalmente se presentan las referencias bibliográficas que apoyaron todos los conceptos desarrollados a lo largo del documento, y algunos anexos que complementan el entendimiento del presente Trabajo Especial de Grado.

CAPÍTULO 1: MARCO CONCEPTUAL

El presente capitulo tiene como objetivo proveer las bases teóricas necesarias para el desarrollo de este Trabajo Especial de Grado, describiendo el contexto bajo el cual se desarrollo el mismo. Este capítulo se divide en tres partes lógicas, en la primera parte se describen los programas de incentivos y su utilidad para las empresas.

En la segunda parte se hace referencia a lo que es una aplicación Web de forma general, así como también su estructura y sus principales características y ventajas. Se explicará la arquitectura Cliente/Servidor, sus características y componentes para luego enumerar sus principales ventajas e inconvenientes, también se explicará la arquitectura de tres capas y finalmente se dará una noción del patrón MVC, que se encuentra dentro de la arquitectura de tres capas, describiendo su flujo de control y principales ventajas.

Finalmente en la tercera parte se describirán las herramientas tecnológicas del lado del cliente como HTML, CCS y JavaScript; se hablará de sus principales características, ventajas y desventajas para posteriormente mencionar la tecnología del lado del servidor ASP.NET utilizada para construir este Trabajo Especial de Grado. También se estudiarán las características, ventajas y desventajas del Sistema Manejador de Base de Datos utilizado para: MSSQL.

1.1. Programas de Incentivos

Los programas de incentivos permiten a las empresas motivar a sus clientes y/o empleados en la compra y/o venta de sus productos mediante la recompensación: cada vez que sus clientes y/o empleados compran o venden alguno de sus productos ganan una cantidad de puntos (o el nombre que le quiera dar la empresa a la moneda) según el producto, estos puntos pueden ser canjeados por premios como bolsos, coolers, sillas de camping, entre otros.

Además de permitir la motivación de los clientes y/o empleados, el programa de incentivos le permite a las empresas conocer la información de mercadeo y logística (en cierta parte) acerca del comportamiento de sus productos en cada una de las diversas sucursales. Esto brinda la oportunidad a cada empresa de conocer tendencias, es decir, perfil de los compradores, qué han comprado, relaciones de las ventas con las actividades calendario, monto promedio de las ventas, etc. para así aplicar inteligencia de negocios en el proceso de toma de decisiones de la alta gerencia de la empresa.

1.2. **Aplicaciones Web** (Dimagin, 2007)

En la ingeniería del software se denomina Aplicación Web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web (los cómputos y procesamiento de la información se realizan en otro computador, llamado servidor) a través de Internet o de una intranet mediante un navegador. El computador, denominado cliente, se comunica con el servidor enviando y recibiendo la información por medio de un navegador web. Un mismo servidor puede interactuar con gran cantidad de clientes al mismo tiempo, por lo cual, todos éstos podrán estar compartiendo los mismos datos y utilizando una misma aplicación desde distintos lugares y sin más requerimientos que un navegador web y una conexión a internet (u otra red compartida).

1.2.1. Características de las Aplicaciones Web (Good, 2005)

Algunas de las características de las Aplicaciones Web son:

Compatibilidad multiplataforma: Las Aplicaciones Web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables. Varias tecnologías incluyendo Java, Flash, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.

Actualización: Las aplicaciones basadas en web están siempre actualizadas con la última versión sin requerir que el usuario tome acciones pro-activas, y sin necesitar llamar la atención del usuario o interferir con sus hábitos de trabajo con la esperanza de que va a iniciar nuevas descargas y procedimientos de instalación.

Inmediatez de acceso: La mayoría de las aplicaciones basadas en web no necesitan ser descargadas, instaladas y configuradas.

Facilidad de prueba: Finalmente no habrá más obstáculos para permitir pruebas sencillas y efectivas de herramientas y aplicaciones antes de realizar una operación que afecta la realidad como por ejemplo cargar la tarjeta de crédito, es decir poder simular este tipo de operaciones.

Menor requerimiento de memoria: Las aplicaciones basadas en web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, esas aplicaciones basadas en web usan en muchos casos la memoria de las computadoras en las que ellas corren, dejando más espacio para correr múltiples aplicaciones al mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.

Datos online: Por supuesto con el desplazamiento de las aplicaciones locales a aquellas basadas en web también los datos que se crean y acceden van a necesitar experimentar profundos cambios porque van a estar en constante movimiento dentro de la red al alcance de todo el mundo, incluso de personas inescrupulosas.

Múltiples usuarios concurrentes: Las aplicaciones basadas en web pueden realmente ser utilizada por múltiples usuarios al mismo tiempo. No hay más necesidad de compartir pantallas o enviar instantáneas cuando múltiples usuarios pueden ver e incluso editar el mismo documento de manera conjunta.

Los datos son más seguros: Si bien la ruptura de discos no va a desaparecer, es probable que los usuarios escuchen mucho menos del tema. A medida que las compañías se hagan cargo del almacenamiento de los datos del usuario, granjas de almacenamiento de datos redundantes, altamente fiables, serán la norma más que la excepción, y los usuarios van a tener mucho menos riesgo de perder sus datos debido a una ruptura de disco impredecible o a un virus de la computadora. Las compañías que proveen aplicaciones basadas en web van a brindar amplios servicios de resguardo de datos ya sea como una parte integral del servicio básico o como una opción paga.

Desarrollar aplicaciones en el lenguaje que se quiera: Una vez que las aplicaciones han sido separadas de computadoras locales y sistemas operativos específicos éstas pueden también

ser escritas en prácticamente cualquier lenguaje de programación. Ya que las aplicaciones web son esencialmente una colección de programas más que un simple programa, ellas podrían ser escritas en cualquier lenguaje de programación que esté por ahí.

1.2.2. Estructura de las Aplicaciones Web

El esquema general de la situación se puede ver en la Figura 1.AW. "Esquema general de las tecnologías Web", donde se muestran cada tipo de tecnología involucrada en la generación e interacción de documentos Web. (Vegas, 2002)

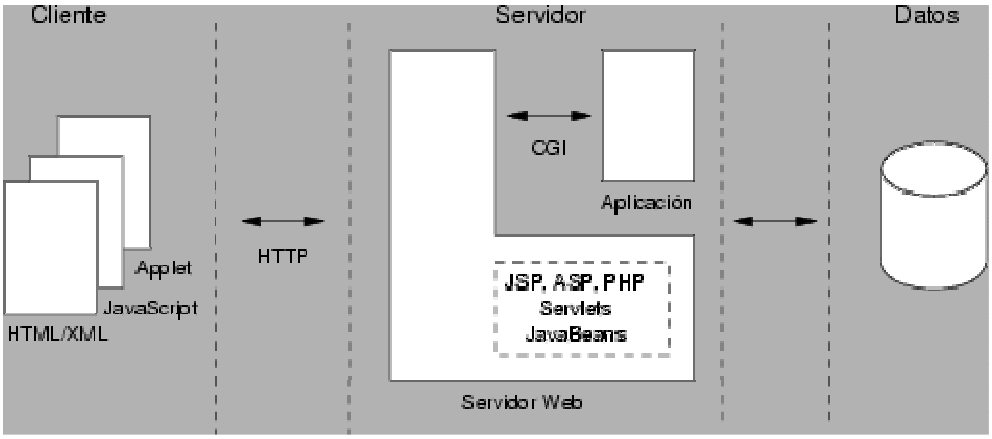

Figura 1.AW. Esquema general de las tecnologías Web (Vegas, 2002)

Aunque existen muchas variaciones posibles, una Aplicación Web está normalmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (ejemplo: PHP, Java Servlets o ASP, ASP.NET, CGI, ColdFusion, embPerl, Python (programming language) o Ruby on Rails) constituye la capa intermedia. Por último, una base de datos constituye la tercera y última capa. (Wikipedia, 2008)

El navegador web manda peticiones a la capa intermedia que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario. (Wikipedia, 2008)

El modo de crear los documentos HTML ha variado a lo largo de la corta vida de las tecnologías Web pasando desde las primeras páginas escritas en HTML almacenadas en un archivo en el servidor Web hasta aquellas que se generan al vuelo como respuesta a una acción del cliente y cuyo contenido varía según las circunstancias. (Vegas, 2002)

Además, el modo de generar páginas dinámicas ha evolucionado, desde la utilización del CGI, *Common Gateway Interface*, hasta los *servlets* pasando por tecnologías tipo *JavaServer Pages*. Todas estas tecnologías se encuadran dentro de aquellas conocidas como *Server Side*, ya que se ejecutan en el servidor web. (Vegas, 2002)

Otro aspecto que completa el panorama son las inclusiones del lado del cliente, *Client Side*, que se refieren a las posibilidades de que las páginas lleven incrustado código que se ejecuta en el cliente, como por ejemplo JavaScript y programas Java. (Vegas, 2002)

En cuanto al servidor de aplicaciones se puede decir que es un software corriendo en una capa intermedia entre un cliente pequeño basado en un explorador y una base de datos. Generalmente se acepta que un servidor de aplicaciones maneja todas las transacciones lógicas y de conectividad que históricamente compartían el cliente y el servidor en un diseño cliente/servidor. La aplicación lógica ha sido movida de clientes grandes y obsoletos a nuevos servidores de aplicaciones como capa intermedia. (Janium, 2007)

1.2.3. Principales ventajas de las Aplicaciones Web (Dimagin, 2007)

Entre las principales ventajas de las Aplicaciones Web se pueden mencionar las siguientes:

- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor.

- No se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador web.

- Se facilita el trabajo a distancia. Se puede trabajar desde cualquier PC o computador portátil con conexión a Internet.

- Actualizar o hacer cambios en el Software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.

- Al funcionar en un navegador, se requiere un conocimiento básico de informática para utilizar una aplicación web.

- La utilización de ésta tecnología conlleva a reducir costos de mantenimiento y complicaciones, y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios.

1.3. Arquitectura Cliente-Servidor

Con respecto a la definición de arquitectura Cliente-Servidor se encuentran las siguientes definiciones:

- Cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada. (Valle & Gutiérrez, 2005)

- Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes'', resultan en un trabajo realizado por otros computadores llamados servidores. (Valle & Gutiérrez, 2005)

- Es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar

información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. (Silice)

1.3.1. Definición de Arquitectura (Valle & Gutiérrez, 2005)

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura Cliente-Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

1.3.2. Definición de Cliente

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. (Valle & Gutiérrez, 2005)

El cliente envía una solicitud al servidor mediante su dirección IP y el puerto, que está reservado para un servicio en particular que se ejecuta en el servidor. (Kioskea)

Los clientes realizan generalmente funciones como: (Silice)

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Entre sus características se encuentran: (Wikipedia, 2008)

- Es quien inicia solicitudes o peticiones, tiene por tanto un papel activo en la comunicación (dispositivo maestro o amo).
- Espera y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.
- Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

Una posible clasificación de los clientes puede ser tomando en cuenta el trabajo que desempeñan, a saber: (Valle & Gutiérrez, 2005)

- "cliente flaco":
  - Servidor rápidamente saturado.
  - Gran circulación de datos de interface en la red.

- "cliente gordo":
  - Casi todo el trabajo en el cliente.
  - No hay centralización de la gestión de la BD.
  - Gran circulación de datos inútiles en la red.

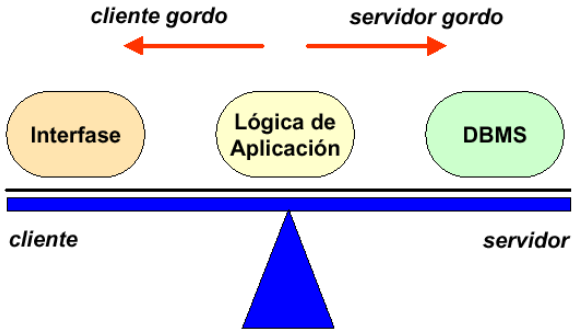

Figura 1.ACS. Balance entre "cliente flaco" y "cliente gordo" (Valle & Gutiérrez, 2005)

1.3.3. Definición de Servidor (Valle & Gutiérrez, 2005)

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN's o WAN's, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

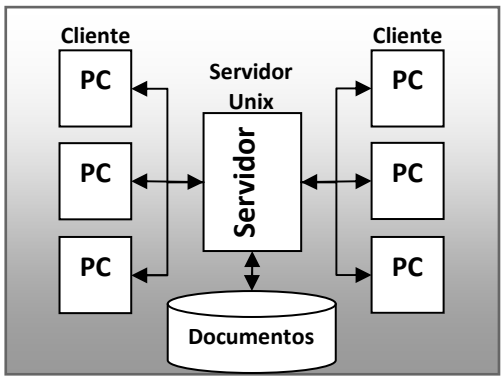

Figura 2.ACS. Ejemplo gráfico de la arquitectura Cliente-Servidor (Valle & Gutiérrez, 2005)

Los servidores realizan generalmente funciones como:

- Espera las solicitudes de los clientes. (Valle & Gutiérrez, 2005)
- Ejecuta muchas solicitudes al mismo tiempo.
- Atiende primero a los clientes VIP.
- Emprende y opera actividades de tareas en segundo plano.
- Se mantiene activa en forma permanente.
- Gestión de periféricos compartidos. (Silice)
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

Entre sus características se encuentran: (Wikipedia, 2008)

- Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).

- No es frecuente que interactúen directamente con los usuarios finales.

Entre los tipos de servidores se encuentran: (Valle & Gutiérrez, 2005)

- *Servidores de archivos*: donde se almacena archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hojas de cálculo, etc.
- *Servidores de bases de datos*: donde se almacenan las bases de datos, tablas, índices. Es uno de los servidores que más carga tiene.
- *Servidores de transacciones*: servidor que cumple o procesa todas las transacciones. Valida primero y genera un pedido al servidor de bases de datos.
- *Servidores de Groupware*: servidor utilizado para el seguimiento de operaciones dentro de la red.
- *Servidores de objetos*: contienen objetos que deben estar fuera del servidor de base de datos. Estos objetos pueden ser videos, imágenes, objetos multimedia en general.
- *Servidores Web*: se usan como una forma inteligente para comunicación entre empresas a través de Internet. Este servidor permite transacciones con el acondicionamiento de un browser específico.

1.3.4. Elementos de la Arquitectura Cliente-Servidor (Valle & Gutiérrez, 2005)

Con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente-Servidor, se deben identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Presentación/Captación de Información.
- Procesos.
- Almacenamiento de la Información.

Los cuales se suelen distribuir tal como se presenta en la Figura 3.ACS. "Aplicaciones Cliente-Servidor":

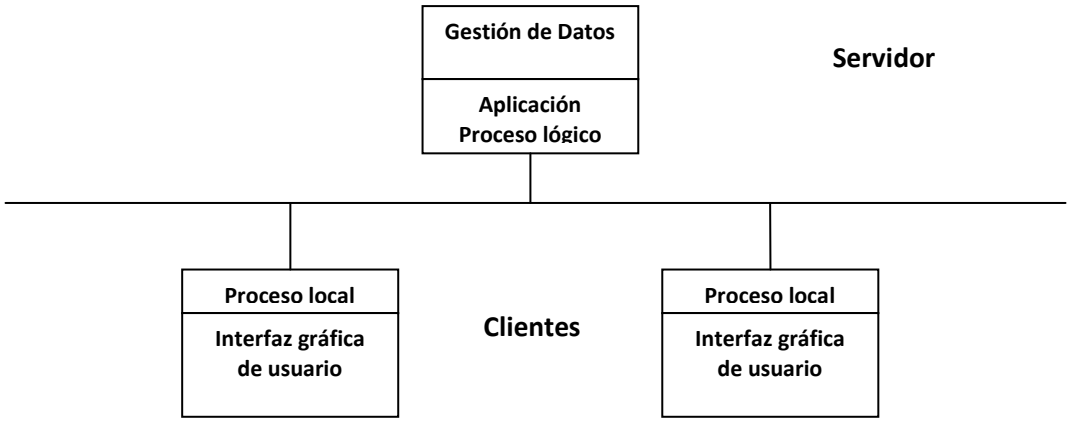

Figura 3.ACS. Aplicaciones Cliente-Servidor (Valle & Gutiérrez, 2005)

Y se integran en una arquitectura Cliente-Servidor en base a los elementos que caracterizan dicha arquitectura, es decir: 1. Puestos de Trabajo, 2. Comunicaciones y 3. Servidores; tal como se presenta en la Figura 4.ACS. “Arquitectura Cliente-Servidor”:

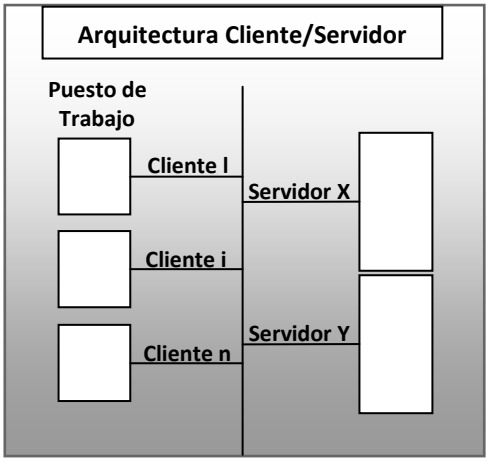

Figura 4.ACS. Arquitectura Cliente-Servidor (Valle & Gutiérrez, 2005)

De estos elementos se pueden destacar:

a. Puesto de Trabajo o Cliente:

Una estación de trabajo o microcomputador (PC: Computador Personal) conectado a una red, que le permite acceder y gestionar una serie de recursos. Se refiere a un microcomputador conectado al sistema de información y en el que se realiza una parte mayoritaria de los procesos.

Se debe destacar que el puesto de trabajo basado en un microcomputador conectado a una red, favorece la flexibilidad y el dinamismo en las organizaciones. Entre otras razones, porque permite modificar la ubicación de los puestos de trabajo, dadas las ventajas de la red.

b. Servidores o Back-End:

Una máquina que suministra una serie de servicios como Bases de Datos, Archivos, Comunicaciones, etc.

Los Servidores, según la especialización y los requerimientos de los servicios que debe suministrar pueden ser: mainframes, miniordenadores o especializados (Dispositivos de Red, Imagen, etc.).

Una característica a considerar es que los diferentes servicios, según el caso, pueden ser suministrados por un único Servidor o por varios Servidores especializados.

c. Comunicaciones

En sus dos vertientes:

- Infraestructura de redes: componentes Hardware y Software que garantizan la conexión física y la transferencia de datos entre los distintos equipos de la red.

- Infraestructura de comunicaciones: componentes Hardware y Software que permiten la comunicación y su gestión, entre los clientes y los servidores.

1.3.5. Características de la Arquitectura Cliente-Servidor (Valle & Gutiérrez, 2005)

En la arquitectura Cliente-Servidor se pueden encontrar las siguientes características:

- El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

- Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

- Un servidor da servicio a múltiples clientes en forma concurrente, presentándoles a todos una interfaz única y bien definida.

- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes**.** Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final. También es importante hacer notar que las funciones Cliente-Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

- Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

En conclusión, Cliente-Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

1.3.6. Ventajas de la Arquitectura Cliente-Servidor

A continuación se desglosan algunas de las ventajas de la arquitectura Cliente-Servidor según aspectos de productividad, costo y rendimiento:

a. Aumento de la productividad: (Silice)

- Mediante la integración de las aplicaciones Cliente-Servidor con las aplicaciones personales de uso habitual, los usuarios pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.

- Una interfaz gráfica de usuario consistente reduce el tiempo de aprendizaje de las aplicaciones.

b. Menores costes de operación:

- Permiten un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. (Silice)

- El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

- Fácil mantenimiento, al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación. (Wikipedia, 2008)

c. Mejora en el rendimiento de la red: (Silice)

- Las arquitecturas Cliente-Servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso.

- Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones.

- La existencia de varias CPU's proporciona una red más fiable: un fallo en uno de los equipos no significa necesariamente que el sistema deje de funcionar.

- La arquitectura modular de los sistemas Cliente-Servidor permite el uso de computadores especializados (servidores de base de datos, servidores de ficheros, estaciones de trabajo para CAD, etc.).

1.3.7. Inconvenientes de la Arquitectura Cliente-Servidor (Silice)

- Hay una alta complejidad tecnológica al tener que integrar una gran variedad de productos.

- Requiere un fuerte rediseño de todos los elementos involucrados en los sistemas de información (modelos de datos, procesos, interfaces, comunicaciones, almacenamiento de datos, etc.).

- Es más difícil asegurar un elevado grado de seguridad en una red de clientes y servidores que en un sistema con un único computador centralizado.

- A veces, los problemas de congestión de la red pueden degradar el rendimiento del sistema por debajo de lo que se obtendría con una única máquina (arquitectura centralizada).

- El quinto nivel de esta arquitectura (bases de datos distribuidas) es técnicamente muy complejo y en la actualidad hay muy pocas implantaciones que garanticen un funcionamiento totalmente eficiente.

1.4. Arquitectura de Tres Capas

La estrategia tradicional de utilizar aplicaciones compactas causa gran cantidad de problemas de integración en sistemas de software complejos como pueden ser los sistemas de gestión de una empresa o los sistemas de información integrados consistentes en más de una aplicación. Estas aplicaciones suelen encontrarse con importantes problemas de escalabilidad, disponibilidad, seguridad, integración... (Sánchez Gónzalez, 2004)

Para solventar estos problemas se ha generalizado la división de las aplicaciones en capas que normalmente serán tres: una capa que servirá para guardar los datos (base de datos), una capa para centralizar la lógica de negocio (modelo) y por último una interfaz gráfica que facilite al usuario el uso del sistema. (Sánchez Gónzalez, 2004)

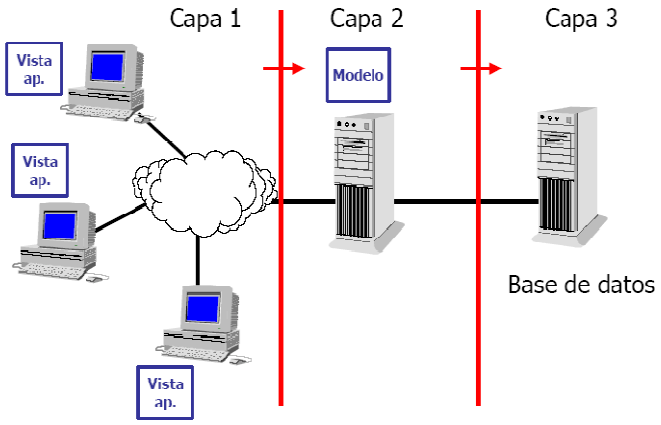

Figura 1.ATC. Arquitectura en Tres Capas (Sánchez Gónzalez, 2004)

Al ser la primera capa un servicio, se puede inferir que las aplicaciones no solo podrían ser de escritorio, si se quisiera que la aplicación tenga una interface web, pues solamente bastaría con cambiar la capa de presentación y de allí en adelante nada tiene porque cambiar. (Loja, 2007)

Esta separación en capas proporciona escalabilidad, capacidad de administración y utilización de recursos mejorados. Cada capa es un grupo de componentes que realiza una función específica. Se puede actualizar una capa sin recompilar otras capas. (Rodríguez Terrero, 2005)

Por regla general, La capa de la presentación es una interfaz gráfica que muestra los datos a los usuarios, siendo responsable de: (Loja, 2007)

- Obtener información del usuario.
- Enviar la información del usuario a los servicios de negocios para su procesamiento.
- Recibir los resultados del procesamiento de los servicios de negocios.
- Presentar estos resultados al usuario.

La capa de la lógica de negocios es responsable de procesar los datos recuperados y enviarlos a la capa de presentación: (Loja, 2007)

- Recibir la entrada del nivel de presentación.
- Interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a automatizar (por ejemplo, la preparación de impuestos por ingresos, el procesamiento de órdenes y así sucesivamente).
- Enviar el resultado procesado al nivel de presentación.

La capa de datos almacena los datos de la aplicación en un almacén persistente, tal como una base de datos relacional o archivos XML y es responsable de: (Loja, 2007)

- Almacenar los datos.
- Recuperar los datos.
- Mantener los datos.
- La integridad de los datos.

Se pueden alojar todas las capas en el mismo servidor, pero también es posible alojar cada capa en varios servidores. (Rodríguez Terrero, 2005)

Las aplicaciones que se construyen con una arquitectura multicapa tienen entre otras las siguientes características: (Loja, 2007)

- **Acceso a bases de datos** (BD): Normalmente con BD relacionales.

- **Transaccionales**: Propiedades ACID (Atomicity-Consistency-Isolation-Durability)
  - *Operaciones atómicas* (Atomicity): son operaciones que se completan en su totalidad o no se completan en absoluto.
  - *Transformaciones consistentes* (Consistency): preservan la integridad interna de los recursos involucrados.
  - *Transformaciones aisladas* (Isolation): parecen ocurrir serialmente, una detrás de otra, creando la ilusión de que ninguna transformación está siendo ejecutada al mismo tiempo.
  - *La durabilidad* (Durability): se refiere a la habilidad para almacenar los resultados de una transformación de estado, usualmente en un disco, de tal modo que los resultados de una transformación puedan ser recuperados en caso de una falla del sistema.

- **Escalables**: Deberían poder soportar más carga de trabajo sin necesidad de modificar el software (sólo añadir más máquinas).

- **Disponibilidad**: Idealmente no deben dejar de prestar servicio.

- **Seguras**: No todos los usuarios pueden acceder a la misma funcionalidad.

- **Integración**: Es preciso integrar aplicaciones construidas con distintas tecnologías.

- **Tipo de interfaz**:
  - De entorno de ventanas (clientes desktop): normalmente sólo tiene sentido en intranets.
  - Web: En Internet y en intranets.

- **Separación clara entre la interfaz gráfica y la capa de componentes**: La capa de componentes encapsula la lógica de negocio. La capa de componentes debería ser reusable con distintas interfaces gráficas.

1.5. Patrón de Diseño MVC (Modelo-Vista-Controlador)

Un patrón de diseño en ingeniería del software es un modelo formal o semiformal que es aplicable a diferentes dominios. Estos problemas aunque de diferentes ámbitos son semejantes desde el punto de vista de la estructura lógica de la solución. El patrón de diseño es el denominador común, una estructura común que tiene diversas aplicaciones. En ingeniería del software es usual distinguir entre dominio del problema y dominio de la aplicación (de la solución). Por tanto, los patrones ayudan a seguir unas pautas comunes en la solución de problemas diferentes, pero semejantes en su estructura. (Gutiérrez Mayoral, 2005)

La arquitectura Modelo-Vista-Controlador surgió como patrón arquitectónico para el desarrollo de interfaces gráficas de usuario en entornos Smalltalk (Sánchez Gónzalez, 2004). El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, entonces trabajando en Smalltalk en laboratorios de investigación de Xerox (Wikipedia, 2008). Su concepto se basaba en separar el modelo de datos de la aplicación de su representación de cara al usuario y de la interacción de éste con la aplicación, mediante la división de la aplicación en tres partes fundamentales: (Sánchez Gónzalez, 2004)

- El modelo, que contiene la lógica de negocio de la aplicación.
- La vista, que muestra al usuario la información que éste necesita.
- El controlador, que recibe e interpreta la interacción del usuario, actuando sobre modelo y vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización.

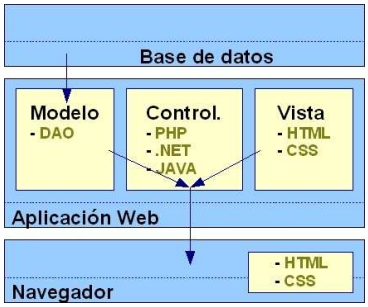

Figura 1.MVC. Patrón MVC (Programación Web, 2007)

El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. (Wikipedia, 2008)

Esta arquitectura de aplicaciones otorga varias ventajas clave al desarrollo de aplicaciones web, destacando:

- Al separar de manera clara la lógica de negocio (modelo) de la vista permite la reusabilidad del modelo, de modo que la misma implementación de la lógica de

negocio que maneja una aplicación pueda ser usado en otras aplicaciones, sean éstas web o no. (Sánchez Gónzalez, 2004)

- Permite una sencilla división de roles, dejando que sean diseñadores gráficos sin conocimientos de programación o desarrollo de aplicaciones los que se encarguen de la realización de la capa vista.

- Cuando se realiza un cambio de bases de datos, programación o interfaz de usuario solo se toca uno de los componentes.

- Se puede modificar uno de los componentes sin conocer cómo funcionan los otros. (Programación Web, 2007)

1.5.1. Descripción del Patrón MVC

Para describir el patrón de diseño MVC es necesario describir cada uno de sus componentes: Modelo, Vista y Controlador.

a. Modelo:

Esta es la representación específica de la información (datos y reglas del negocio) con la cual el sistema opera, es decir modela los datos y el comportamiento detrás de los procesos de negocio (Wikipedia, 2008). La lógica de datos asegura la integridad de estos, ya que permite generar restricciones, y también permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra. Es preferible que el modelo sea independiente del sistema de almacenamiento de datos (Gutiérrez Mayoral, 2005).

Es responsable de realizar: (González & Machuca, 2008)

- Consultas a la base de datos.
- Ejecutar los cálculos de los procesos de negocio.
- Procesar órdenes.
- Encapsular los datos y el comportamiento que son independientes de la presentación.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo. (Wikipedia, 2008)

b. Vista:

Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario (Wikipedia, 2008). Define la forma de mostrar la información al usuario. Normalmente, tiene un registro del controlador asociado. (Gutiérrez Mayoral, 2005)

Despliega los resultados de la lógica de negocios (modelo). No está relacionada en como la información fue obtenida o de donde (pues eso es responsabilidad del modelo). (González & Machuca, 2008)

c. Controlador:

Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista (Wikipedia, 2008). El controlador es el encargado de redirigir un procesamiento determinado para cada petición que reciba, es decir, sirve como una conexión lógica entre la interacción del usuario y los servicios de negocio disponibles. Por tanto, el controlador debe poseer algún modo de conocer la correspondencia entre peticiones y posibles respuestas; deberá tener un mapa de peticiones y respuestas. (González & Machuca, 2008)

Es responsable de tomar decisiones acerca de múltiples presentaciones, por ejemplo el lenguaje del usuario o el nivel del acceso del usuario que especifica una presentación distinta. Un request entra a la aplicación a través de la capa de control, el decidirá como el request debe ser manejado y que información debe ser retornada. (González & Machuca, 2008)

1.5.2. Flujo de Control en MVC

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente: (Wikipedia, 2008)

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo (decide quién lleva a cabo la petición en la capa del modelo), actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. Una vez que el modelo termina las operaciones pertinentes, devuelve el control de ejecución al controlador. (Gutiérrez Mayoral, 2005)

5. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. *Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.*

6. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

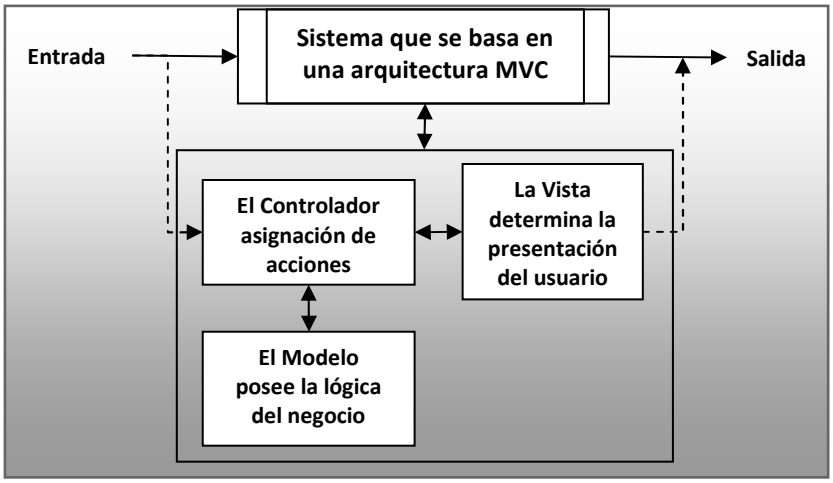

Figura 2.MVC. El esquema de un sistema basado en el patrón software MVC (Gutiérrez Mayoral, 2005)

1.5.3. Ventajas del Modelo MVC (Gutiérrez Mayoral, 2005)

El modelo-vista-controlador para el desarrollo de aplicaciones Web presenta varias ventajas, entre ellas:

- Menor acoplamiento:
  - Desacopla las vistas de los modelos
  - Desacopla los modelos de la forma en que se muestran e ingresan los datos
- Mayor cohesión:
  - Cada elemento del patrón está altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio)
- Las vistas proveen mayor flexibilidad y agilidad:
  - Se puede crear múltiples vistas de un modelo
  - Se puede crear, añadir, modificar y eliminar nuevas vistas dinámicamente
  - Las vistas pueden anidarse
  - Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual
  - Se puede sincronizar las vistas
  - Las vistas pueden concentrarse en diferentes aspectos del modelo
- Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales:
  - Una vista para cada dispositivo que puede variar según sus capacidades
  - Una vista para la Web y otra para aplicaciones de escritorio
- Más claridad de diseño.
- Facilita el mantenimiento.
- Mayor escalabilidad.

1.6. Tecnologías del Lado del Cliente

Las tecnologías del lado del cliente están embebidas en la página HTML y son interpretadas y ejecutadas por el navegador Web, entre estas se pueden mencionar las siguientes:

1.6.1. HTML

HTML (*HyperText Markup Language*) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con *enlaces* (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con *inserciones* multimedia (gráficos, sonido, ...). La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado (como Mosaic, o Netscape). (Martínez Echeverría, 1995)

HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (Wikipedia, 2008)

HTML también es usado para referirse al contenido del tipo MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores). (Wikipedia, 2008)

El HTML fue creado en 1986 por el físico nuclear Tim Berners-Lee; el cual tomo dos herramientas preexistentes: el concepto de Hipertexto (conocido también como link o ancla) el cual permite conectar dos elementos entre sí y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no presenta ningún compilador, por lo tanto algún error de sintaxis que se presente éste no lo detectará y se visualizara en la forma como éste lo entienda. (Ravioli)

El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html. Estos documentos pueden ser mostrados por los visores o "browsers" de páginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer. (Ravioli)

También existe el HTML Dinámico (DHTML), que es una mejora de Microsoft de la versión 4.0 de HTML que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página. (Ravioli)

1.6.2. CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. (Eguiluz Pérez, 2008)

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Eguiluz Pérez, 2008)

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. (Eguiluz Pérez, 2008)

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc. (Eguiluz Pérez, 2008)

Entre las ventajas de utilizar CSS (u otro lenguaje de estilo) están: (Wikipedia, 2008)

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea).

Entre las desventajas de utilizar CSS están: (Wikipedia, 2008)

- No todos los navegadores soportan las Hojas de Estilos (solo versiones 4 o superiores de Explorer y Netscape).

- No existe un estándar general para todos los navegadores, por lo que el resultado del CSS difieren de un navegador a otro.

1.6.3. JavaScript (Eguiluz Pérez, 2008)

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios, lo que implica que si el navegador no soporta JavaScript, no se podrán ejecutar las funciones que se programen. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

1.7. Tecnología del Lado del Servidor

Una tecnología del lado del servidor es aquella que se ejecuta en el servidor *Web*, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. Como la página resultante contiene sólo código HTML, es compatible con todos los navegadores. (Wikipedia, 2008)

1.7.1. ASP.NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Las "Páginas Activas" se utilizan para ejecutar acciones del lado del servidor. De forma opuesta al Javascript, que realiza procedimientos en la máquina de cada usuario, el ASP forma en el servidor los resultados que luego se mostrarán en las pantallas de cada navegante. Todo este procedimiento se realiza en el servidor y no en la máquina. (Papyros Digitales)

Las páginas activas, o dinámicas, son especialmente útiles para mantener bases de datos, crear buscadores dinámicos, hacer carritos de compras, y todo aquello que necesite una interacción del navegante y el servidor para elaborar un resultado. (Papyros Digitales)

Las páginas ASP pueden hacer uso de objetos COM (*Component Object Model*) que son objetos en algún otro lenguaje (ej.: ejecutables en C++ o Java); de manera que si ya se tiene algo programado, las páginas ASP a través del Internet Information Server pueden hacer uso de los métodos en estos objetos. Sin embargo, su modelo de desarrollo es más bien funcional, y no totalmente orientado a objetos. (Castillo)

Para conectarse a una base de datos, normalmente se utiliza ADO, ya sea por ODBC (Conectividad de Bases de Datos Abiertas) u OLEdb (Empotramiento y Vinculación de Objetos de BD), que es un adaptador universal a bases de datos que se especializa posteriormente para hablar con una base de datos concreta. (Castillo)

El esquema de trabajo es crear objetos COM que ejecutan la lógica de la aplicación (*Business Objects*) y luego hacer la capa de interfaz con ASP. (Castillo)

1.7.2. Arquitectura ASP.NET (Ercoli, 2007)

Una aplicación web creada con ASP.NET de la forma en la que Microsoft la ha ideado, implica una arquitectura de 2 capas como la siguiente:

Capa	Propósito	Herramientas de software
Aplicación	Interfaz de usuario + lógica de las reglas y procesos del negocio ó dominio.	Páginas web ASP .NET: WebForm (.aspx) + archive code-behind (aspx.cs ó vb)
Acceso a datos	Responsable de mantener la integridad de la BD. Incluye, normalmente, toda la lógica del manejo de transacciones.	Servidor de Base de datos (SqlServer ó cualquier otro)

Tabla 1.ASP. Arquitectura ASP.NET

En el caso de la capa de aplicación, ASP.NET 2.0 no requiere que se ubique necesariamente código de la lógica de la aplicación en el archivo *code-behind*, ya que también se puede agregar el mismo en el código de presentación (.aspx). De todas formas siempre es aconsejable mantener separado las lógicas de aplicación y presentación.

En el caso de la capa de acceso a datos, no es aconsejable acceder desde la capa de aplicación directamente a la BD (ya sea a través del llamado a procedimientos almacenados ó vía ejecución de comandos SQL), siempre es mejor crear clases que accedan a la tecnología usada en el acceso a datos, ya que con esta decisión de diseño, se obtienen las siguientes ventajas:

- El código de acceso a datos se aísla en una clase separada, por lo cual se puede poner a trabajar a los expertos en BD, mientras los analistas y programadores de lógica de aplicación trabajan en la otra capa.

- Se puede afinar la perfomance de la BD (tiempos de acceso, pruebas de stress, etc.) sin afectar el resto de la aplicación con nuestros cambios.

- Se puede migrar de tecnología de acceso a datos, por ejemplo de SqlServer a Oracle, ó a un ORM (Nhibernate ó Linq, etc.) sin afectar las otras capas (capas como módulos pluggables).

Figura 1.ASP. Aplicación Web ASP .NET (Ercoli, 2007)

1.7.3. Integración con Internet Information Server (IIS) (Ercoli, 2007)

Si se considera la posibilidad de utilizar la autenticación de ASP.NET, es importante comprender cómo interactúa con los servicios de autenticación de IIS. IIS supone siempre que se asigna un conjunto de credenciales a una cuenta de Microsoft Windows NT y utiliza las credenciales para autenticar los usuarios. Existen tres tipos distintos de autenticación disponibles tanto en IIS 5.0 como en IIS 6.0: básica, implícita y Autenticación de Windows integrada (NTLM o Kerberos). Se puede seleccionar el tipo de autenticación que se desea utilizar en los servicios de administración de IIS.

Si se solicita una dirección URL que contiene una aplicación ASP.NET, la información sobre la autenticación y la petición se entrega a la aplicación. ASP.NET proporciona dos tipos adicionales de autenticación, que se describen en la tabla siguiente:

Proveedor de autenticación	Descripción
Autenticación mediante formularios	Sistema que redirige las peticiones no autenticadas a un formulario HTML mediante el redireccionamiento del cliente HTTP. El usuario proporciona las credenciales y envía el formulario. Si la aplicación autentica la petición, el sistema emite un vale de autenticación en una cookie que contiene las credenciales, o una clave, para readquirir la identidad. Las peticiones posteriores se emiten con la cookie en los encabezados de la petición; se autentican y autorizan en un controlador ASP.NET mediante el método de validación que especifique el programador de la aplicación.
Autenticación mediante pasaporte	Servicio de autenticación centralizado proporcionado por Microsoft que ofrece a los sitios Web suscritos servicios de perfil básico y un inicio de sesión único.

Tabla 2.ASP. Tipos de autenticación en ASP.NET

1.7.4. Ventajas de ASP.NET

ASP.NET ofrece varias ventajas importantes sobre los modelos previos de desarrollo para Internet: (Manuales PDF, 2007)

- *Mejor eficiencia*: ASP.NET corre código compilado sobre el entorno NGWS en el servidor. Distinto a sus predecesores interpretados, ASP.NET usa amarres tempranos ("early binding"), así como compilación justo a tiempo ("just-in-time compilation"), optimización nativa, y servicios de caché, sin configuración adicional. Para los desarrolladores, esto significa eficiencia dramáticamente superior antes de escribir la primera línea de código.

- *Herramientas superiores de desarrollo*: ASP.NET tiene una "caja de herramientas" rica: el ambiente de desarrollo integrado de Visual Studio .NET. La edición WYSIWYG, la creación de controles mediante "drag-and-drop", y la publicación automática son varias ventajas.

- *Poder y flexibilidad*: Porque ASP.NET está basado en el Entorno Común de Ejecución de Lenguajes (Common Language Runtime, o "CLR") el poder y la flexibilidad de la plataforma completa está disponible para los desarrolladores. Las librerías de Clases del CLR, la Mensajería, y las soluciones de Acceso a Datos, son accesibles a través del Internet. ASP.NET permite el uso de una gran variedad de lenguajes de programación y, por tanto, se puede escoger el mejor lenguaje para la aplicación, o particionarla en varios lenguajes. Más aún, la interoperabilidad del CLR garantiza que la inversión en el desarrollo de aplicaciones COM sea preservada cuando se migra a ASP.NET.

- *Simplicidad*: ASP.NET hace fácil el ejecutar tareas comunes, desde el simple envío de un formulario o la autenticación de un cliente, hasta el despliegue y la configuración de una Web. Por ejemplo, el entorno de paginado de ASP.NET permite construir interfaces de usuario que separan limpiamente la lógica de la aplicación del código de la presentación, y maneja eventos con un modelo sencillo de procesamiento de formularios al estilo de Visual Basic. Adicionalmente, el CLR simplifica el desarrollo con servicios de código gerenciado, como el conteo automático de referencias y la limpieza automática de la memoria utilizada por su aplicación.

- *Gerenciabilidad*: ASP.NET usa un sistema jerárquico de configuración, basado en archivos de texto, que simplifica la aplicación de parámetros de configuración al servidor y sus aplicaciones. Porque la información de configuración es almacenada como texto, nuevos parámetros pueden ser configurados sin recurrir a herramientas de administración locales. Una aplicación de ASP.NET se despliega a un servidor simplemente copiando los archivos necesarios al servidor. No hay que reiniciar el servidor, ni siquiera para reemplazar código compilado que ya está en servicio.

- *Escalabilidad y disponibilidad*: ASP.NET ha sido diseñado para la escalabilidad con características específicamente dirigidas a mejorar el funcionamiento de servidores racimados (clustered) y de servidores con procesadores múltiples. Los procesos del servidor son vigilados y gerenciados por el entorno del ambiente de ejecución de ASP.NET, así que si algún proceso se entorpece o se detiene, un nuevo proceso puede

ser creado para reemplazarlo, lo cual ayuda a mantener la disponibilidad de su aplicación para manejar solicitudes de servicio.

- *Personalización y extensibilidad*: ASP.NET entrega una arquitectura bien formada que permite que los desarrolladores "enchufen" su código al nivel apropiado. De hecho, es posible el extender o reemplazar cualquier sub-componente del ambiente de ejecución de ASP.NET con un componente personalizado. La implementación de autenticación personalizada o de servicios de mantenimiento de estado nunca ha sido tan sencilla.

- *Seguridad*: Con autenticación nativa de Windows y configuración individual por aplicación: las aplicaciones están seguras.

La plataforma Microsoft .NET ofrece actualmente compatibilidad integrada para lenguajes como: C#, C++, Visual Basic, J# y JScript. (Microsoft Corporation, 2001)

En cuanto a los servidores, el ASP es una tecnología de Microsoft y para poder trabajar se necesita tener instalado uno de estos dos programas: (González Estrada)

- IIS (Internet Information Server)
- PWS (Personal Web Server)

Otra área importante en los preparativos para llevar a cabo una migración es el acceso a datos. La aparición de ADO .NET proporciona una forma nueva y eficaz de obtener acceso a los datos. En la mayoría de los casos, se puede continuar usando ADO como en el pasado, pero es muy recomendable echar un vistazo a ADO .NET a fin de mejorar los métodos de acceso a datos en la aplicación ASP.NET. (González Estrada)

1.8. Sistema Manejador de Base de Datos

Un SMBD consiste de un conjunto de datos relacionados entre sí y un conjunto de herramientas de software (y/o hardware) para tener acceso a esos datos. Un SMBD puede organizar, procesar y presentar los datos seleccionados de una BD. Esta capacidad permite a quienes toman decisiones rastrear, probar y consultar el contenido de la BD para extraer las respuestas a las preguntas no recurrentes y no previstas en informes regulares.

Los SMBD administrarán los datos almacenados y reunirán las partes necesarias de la BD común para responder a las preguntas de quienes no sean programadores.

El objetivo primordial de un SMBD es proporcionar un entorno para recuperar información y almacenar nueva información en la BD, para lo cual debe proporcionar a los usuarios una visión abstracta de los datos. Es decir, los detalles de cómo se almacenan y se mantienen los datos, son transparentes para los usuarios. Esto se debe a que muchos de ellos, no tienen experiencia en computadores, por ello se les esconde la complejidad a través de diversos niveles de abstracción, para simplificar la interacción con el sistema.

1.8.1. MSSQL (SQL Server)

SQL Server es un sistema manejador de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la

base de datos SQL Server ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento. (Microsoft, 2008)

El motor de datos SQL Server constituye el núcleo de esta solución de administración de datos empresariales. Asimismo, SQL Server combina lo mejor en análisis, información, integración y notificación. (Microsoft, 2008)

La integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business Intelligence Development Studio, distingue a SQL Server. La administración de MSSQL 2005 es simple con una GUI agradable. Casi cualquier persona experimentada de MSSQL puede realizar el papel del DBA con éxito. (Microsoft, 2008)

Una de las características realmente agradables y útiles proporcionadas por MSSQL 2000/2005 es los servicios integrados de la información que incluye la ayuda de la creación del pdf y un IDE agradable. Los servicios de información del MSSQL 2005 son integrados en el RDBMS. (Microsoft, 2008)

Para los servidores del nivel de departamento o sistemas funcionando en máquinas de la gama pequeña/mediana el servidor de MSSQL sería una buena opción. (Microsoft, 2008)

La característica más notable agregada en MSSQL 2005 es el nuevo nivel de aislamiento llamado Snapshot Isolation (SI). La idea ha sido agregar la opción versioning de la fila a MSSQL. (Microsoft, 2008)

MSSQL 2005 tiene dos opciones de exportación e importación, a saber BCP y DTS. BCP puede exportar e importar datos usando un formato textual portable que se puede importar fácilmente al RDBMS de otro vendedor. DTS tiene un alto rendimiento, una herramienta programable de ETL con tareas y proporciona la opción para construir el flujo de trabajo similar a los construidos por el constructor de Oracle Workflow. Sin embargo DTS se utiliza sobre todo para poblar almacenes de datos más que para construir flujo de negocio. (Microsoft, 2008)

MSSQL cuenta con la seguridad basada en roles para el servidor, bases de datos y perfiles de aplicación, herramientas integradas para la seguridad de auditoría, seguimiento de 18 diferentes eventos de seguridad adicionales y sub-eventos, más apoyo para el archivo y sofisticada red de cifrado, incluyendo SSL, Kerberos y delegación. MSSQL 2000 ha sido certificado por el gobierno federal del nivel C2 - la certificación de la seguridad - el más alto nivel de seguridad disponible en la industria. (Dichiara, 2004)

El siguiente diagrama ilustra los componentes básicos en SQL Server, muestra cómo SQL Server es una parte importante de Windows Server System y se integra con la plataforma Microsoft Windows, incluidos Microsoft Office System y Visual Studio. (Microsoft, 2008)

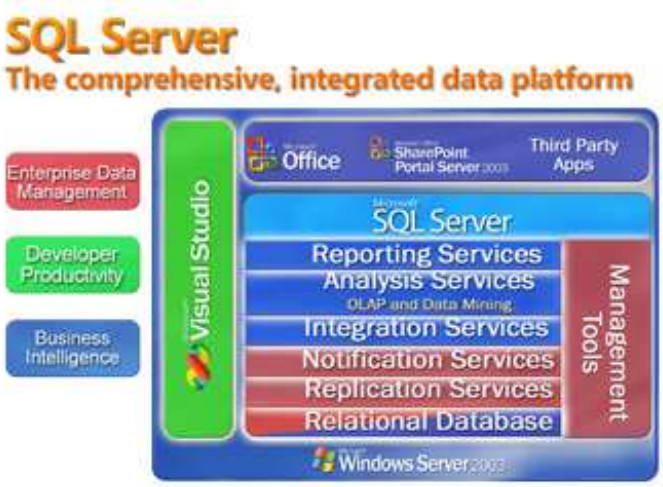

Figura 1.MSSQL. Componentes básicos de SQL Server (Microsoft, 2008)

1.8.2. Características de Microsoft SQL Server (Gonzalez Urmachea)

Entre las características más resaltantes del manejador de base de datos SQL Server se encuentran:

- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- *Escalabilidad*: Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles.
- *Potencia*: Microsoft SQL Server es la mejor base de datos para Windows NT Server. Posee los mejores registros de los benchmarks independientes (TCP) tanto en transacciones totales como en coste por transacción.
- *Orientada al desarrollo*: Visual Basic, Visual C++, Visual J++, Visual Interdev, Microfocus Cobol y muchas otras herramientas son compatibles con Microsoft SQL Server.
- Diseñada desde su inicio para trabajar en entornos Internet e Intranet, Microsoft SQL Server es capaz de integrar los nuevos desarrollos para estos entornos específicos con los desarrollos heredados de aplicaciones "tradicionales". Es más, cada aplicación desarrollada para ser empleada en entornos de red local puede ser utilizada de forma transparente -en parte o en su totalidad- desde entornos Internet, Intranet o Extranet.
- Diseñada para INTERNET: Es el único gestor de base de datos que contiene de forma integrada la posibilidad de generar contenido HTML de forma automática.
- Arquitectura de servidor simétrico y paralelo con balanceo automático de carga en múltiples procesadores.
- Kernel multithread real para mejor rendimiento transaccional y escalabilidad.
- Soporte grandes bases de datos (VLDB) (+1 TB).
- Completo proceso transaccional interactivo con rollback automático y recuperación de roll-forward.
- Optimizador de consultas mejorado basado en coste.
- Checkpointing mejorado para un mejor throughput de datos y tiempo de respuesta.
- Soporte E/S asíncrono para acceso en paralelo a múltiples dispositivos de disco para un mejor throughput.
- Bloqueo a nivel fija y página con escalación de bloqueos; resolución automática de deadlocks.

- Llamadas a procedimientos remotos servidor-a-servidor (procedimientos almacenados remotos).
- Replicación asíncrona o contínua basada en registros, o sincronización planificada de tablas point-in-time.
- Configuración de replicación gráfica y características de gestión.
- Replicación de subscriptores ODBC, incluyendo IBM DB2, ORACLE, SYBASE y Microsoft Access.
- El Distributed Transaction Coordinator gestiona transacciones que involucran a dos o más servidores SQL (proceso Two Phase Commit 2PC) transparente.
- Replicación de tipos de datos Texto e Imagen.
- Triggers, procedimientos almacenados (autoexec), disparador de eventos antes y después de conexiones.
- Procedimientos almacenados extendidos (funciones definidas por el usuario) utilizando C/C++.
- Cursores basados en el motor con scrolling hacia adelante y atrás; posicionamiento absoluto y relativo.
- Sentencias DLL permitidas dentro de transacciones.
- Transacciones distribuidas dentro de interfaces DB-Librery, ODBC, Transact-SQL, XA y OLE Transaction.
- Procedimientos almacenados OLE Automation.

CAPÍTULO 2: MARCO METODOLÓGICO

Planificar y ejecutar un proyecto que beneficie a una empresa u organización requiere de una metodología que dote de los mecanismos adecuados para que el sistema satisfaga las necesidades tanto de los usuarios como de los clientes que contratan dicho desarrollo. De nada sirven buenas notaciones y herramientas si no se proveen directivas para su aplicación. Así, en estos últimos años se ha comenzado con un creciente interés en metodologías de desarrollo, que permitan obtener aplicaciones, mediante un proceso de tratamiento en capas, suficientemente usables y fáciles de mantener. Desde el punto de vista de la ingeniería del software no basta el conocimiento de lenguajes y técnicas de programación, de entornos de desarrollo o de editores de recurso, es imprescindible conocer la manera en la que se debe realizar un software.

La experiencia ha comprobado que un enfoque metódico para el desarrollo de software arroja como resultados menos defectos, reduciendo los tiempos de entrega y agregando valor.

Una metodología plantea un proceso disciplinado con el objetivo de hacer el proceso de desarrollo de software más predictible y eficiente. Alrededor de cómo hacer software hay un gran número de teorías, propuestas y disciplinas de desarrollo, el siguiente apartado tiene como objetivo el estudio de algunas de ellas como lo son Scrum y RUP (Rational Unified Process) cuyas implementaciones sirven de guía para la elaboración de este Trabajo Especial de Grado, analizando sus principales características y describiendo las etapas que contempla cada una.

2.1. Scrum

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores e impulsores de metodologías de software.
El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”. (Gúzman Matus)

Según el Manifiesto Ágil se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas: Es más importante construir un buen equipo que construir el entorno.
- Desarrollar software que funciona más que conseguir una buena documentación: No producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante.
- La colaboración con el cliente más que la negociación de un contrato: Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- Responder a los cambios más que seguir estrictamente un plan: Se debe ser hábil en responder a los cambios y a los fracasos, la planificación no debe ser estricta sino flexible y abierta.

Uno de los representantes de las metodologías ágiles, basada en el Manifiesto Ágil es Scrum.

Scrum es una forma de gestionar proyectos de software. No es una metodología de análisis, ni de diseño, como podría ser RUP, es una metodología de gestión del trabajo. Una de las características más importantes es que es muy fácil de explicar y de entender, lo que ayuda mucho a su implantación. (Gracia, 2006)

Por otra parte SCRUM puede ser aplicado a distintos modelos de calidad (como podría ser CMMI) puesto que estos dicen qué se tiene que hacer, es decir, dicen que se gestiona pero no dicen cómo. Ahí es donde entra SCRUM como modelo de gestión del proyecto. (Gracia, 2006)

Scrum debe todas sus prácticas desde un proceso iterativo e incremental. El esqueleto de Scrum se muestra en la Figura 1.SCRUM. "Esqueleto de SCRUM". El círculo inferior representa una iteración del desarrollo de las actividades que ocurren una tras otra. El producto de cada iteración es un incremento en el producto. El círculo superior representa la reunión diaria que ocurre durante la iteración, en la cual los miembros individualmente del grupo conocen, inspeccionan las actividades y hacen los cambios apropiados. Como resultado de la iteración queda una lista de requerimientos. Este ciclo se repite durante todo el proyecto. (Serrano, 2007)

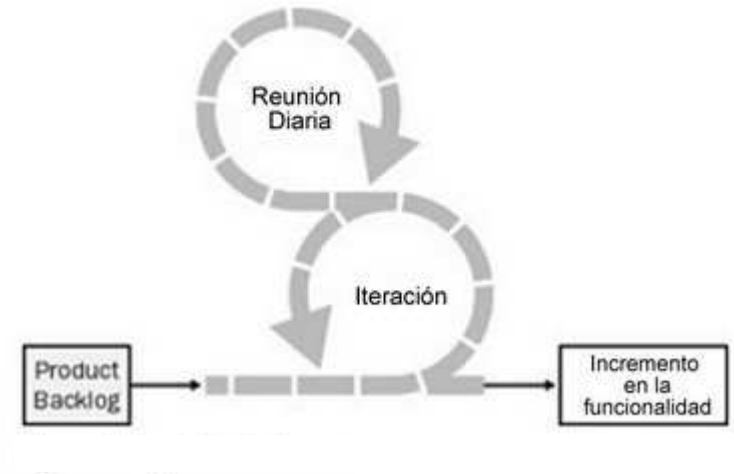

Figura 1.SCRUM. Esqueleto de SCRUM (Serrano, 2007)

Este esqueleto opera de la siguiente manera: (Serrano, 2007)

1. Al comienzo de la iteración, el equipo revisa qué es lo que debe hacer.
2. Luego, selecciona lo que cree que puede hacer para tener un incremento y un potencial prototipo funcional al término de la iteración.
3. El equipo se separa y hace su mejor esfuerzo por el resto de la iteración. Cuando ésta termina, el equipo presenta el incremento de la funcionalidad que construyó, de manera que los otros miembros del equipo puedan revisar las funcionalidades y hacer las modificaciones oportunamente al proyecto.

El equipo revisa los requerimientos, considerando la tecnología disponible, evaluando sus habilidades y capacidades. Luego, determina colectivamente cómo van a construir la funcionalidad, mientras que encuentran y discuten nuevas complejidades, dificultades y sorpresas. El equipo muestra cuáles son las necesidades y cuál es la mejor forma de satisfacerlas. Este proceso de creatividad es el corazón de la productividad de Scrum. (Serrano, 2007)

En Scrum se van a diferenciar dos aspectos importantes, los actores y las acciones. (Serrano, 2007)

2.1.1. Actores de Scrum (Serrano, 2007)

Los actores son los que ejecutarán obviamente las acciones. Los roles se dividen en dos: los comprometidos y los implicados.

Los comprometidos son: el Product Owner o dueño del producto, el ScrumMaster o facilitador, y el equipo.

El papel de los implicados lo juegan: los usuarios del producto o aplicación (quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades), los clientes y vendedores, y los gestores y directivos.

- *El Product Owner (dueño del producto)*: El Product Owner es el responsable de cuidar los intereses de cada uno de los participantes. Representa la voz del cliente y aporta la visión de negocio. Él se encarga de escribir las historias de usuario, les da prioridad y las ubica en la lista de requisitos del producto. El *Product Owner* estima el financiamiento inicial y el requerido en el curso del proyecto.

- *El Team*: Los equipos auto-suficientes, auto-organizados y funcionales, tienen la responsabilidad, en cada iteración, de transformar el Product Backlog en un incremento de la funcionabilidad del producto y planificar su propio trabajo para lograrlo. Lo ideal es que incluya entre 5 y 9 miembros, y que pertenezcan a diferentes disciplinas (desarrolladores, diseñadores, etc.).

- *El ScrumMaster*: El ScrumMaster es responsable del proceso Scrum, debe enseñar la metodología Scrum a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno sigua las reglas y prácticas de Scrum. Tiene como principal papel el de dejar el camino libre de obstáculos e impedimentos para que el resto del equipo consiga el objetivo del sprint.

La gente que sigue estos roles son la personas que confían en el éxito del proyecto. Otros pudieron estar interesados en el proyecto, pero no están comprometidos. Scrum hace una distinción clara entre estos dos grupos y asegura que los que son responsables del proyecto tengan la autoridad suficiente para hacer lo que consideren necesario para el éxito del proyecto y que los que no sean responsables no interfieran innecesariamente. Esta distinción es importante en Scrum. Debe siempre estar clara quién está comprometido y quién está solo implicado.

2.1.2. Acciones de Scrum

Las acciones tienen relación directa con los actores. En Scrum se indican claramente las acciones a acometer y como acometerlas, en la Figura 2.SCRUM. “Acciones de Scrum” se resumen. Debe hacerse siempre de una forma adecuada y algo rígida para impedir que se aplique erróneamente esta metodología. (Serrano, 2007)

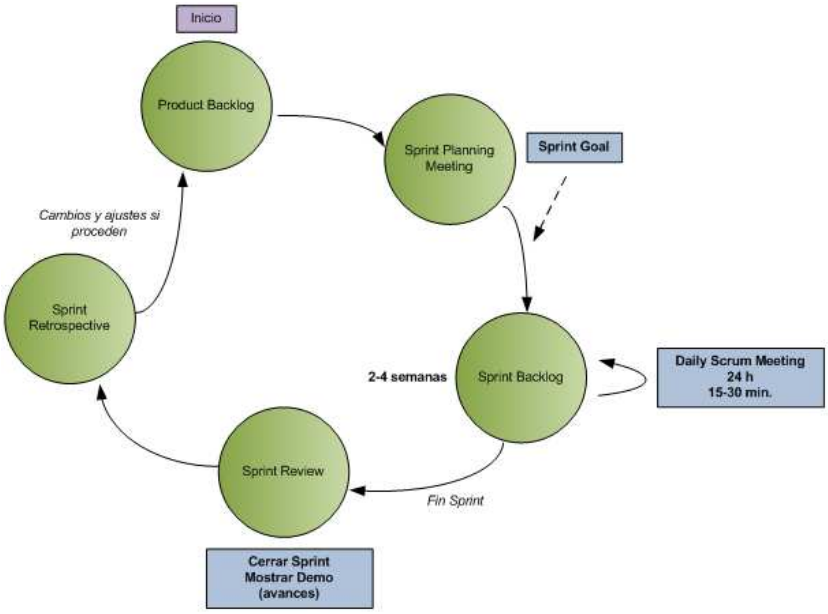

Figura 2.SCRUM. Acciones de Scrum (Serrano, 2007)

- El Product Backlog: (Gúzman Matus)

Un proyecto de Scrum comienza con una visión del sistema que se irá desarrollando a medida que este avance. El dueño del producto formula un plan de modo que se incluya el Product Backlog, el cual va a ser mantenido y actualizado por él mismo. El Product Backlog es una lista de los requisitos funcionales y no funcionales que, cuando se esté pensando en la funcionalidad, debe entregar el camino a seguir. Se da la prioridad al Product Backlog de modo que los artículos que probablemente sirvan para generar valor sean prioridad superior. Los cambios en el Product Backlog reflejan requisitos del negocio que cambian, y cómo el equipo puede transformar rápidamente o lentamente el Product Backlog.

El Product Backlog nunca se acaba, y el Product Backlog usado en la planificación del proyecto, es simplemente una estimación inicial de los requisitos. El Product Backlog se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona. El Product Backlog es dinámico; maneja constantemente los cambios para identificar que necesita el producto para ser: apropiado, competitivo, y útil. Mientras exista un producto, el Product Backlog también existe.

Id	Módulo	Descripción	Est.	Por
Crítico				
1		Plataforma tecnológica	30	AR
2	Cliente	Interfaz de usuario	40	LR
3	Cliente	Un usuario se registra en el sistema	40	LR
4	Trastienda	El operador define el flujo y textos de un expediente	60	AR
5	Trastienda	Etc...	999.	XX
Necesario				
6	Cliente	El usuario modifica su ficha personal	30	AR
7	Cliente	El usuario consulta los expedientes asignados	15	LR
8	Cliente	El usuario tramita un expediente	35	LR

Figura 3.SCRUM. Ejemplo de Product Backlog (Gúzman Matus)

- Sprint Backlog: (Gúzman Matus)

Todo el trabajo se hace en Sprint. Cada Sprint es una iteración que suele realizarse en un plazo de entre 2 y 4 semanas, esto debe ser marcado antes de iniciar el Sprint Backlog, de hecho, del Product Backlog se sacará la tarea o tareas realistas para acometer el Sprint Backlog. Una norma fundamental es que mientras un Sprint Backlog se inicia, éste **NO** puede ser alterado o modificado. Hay que esperar a que concluya el Sprint Backlog para realizar la correspondiente modificación o alteración cuya tarea, formaría parte de otro Sprint Backlog. Al final, el objetivo es entregar algo que funcione, para que el usuario pueda probarlo y se puedan introducir los cambios necesarios antes de que sea demasiado tarde. Esto es lo que permitirá ser flexibles. Cada Sprint se inicia con un Sprint Planning Meeting (reunión de planeamiento del Sprint). El Sprint Backlog define el trabajo, o las tareas, que el Team desarrollará para poder convertir el Product Backlog seleccionado para ese Sprint, en un incremento potencialmente funcional del producto. Las tareas deben ser divididas de modo que cada una demore entre 4 a 16 horas finalizarlas. Las tareas de largo mayor de 16 horas se consideran secundarias, ya que todavía no se han definido apropiadamente. Solamente el equipo puede cambiar el Sprint Backlog.

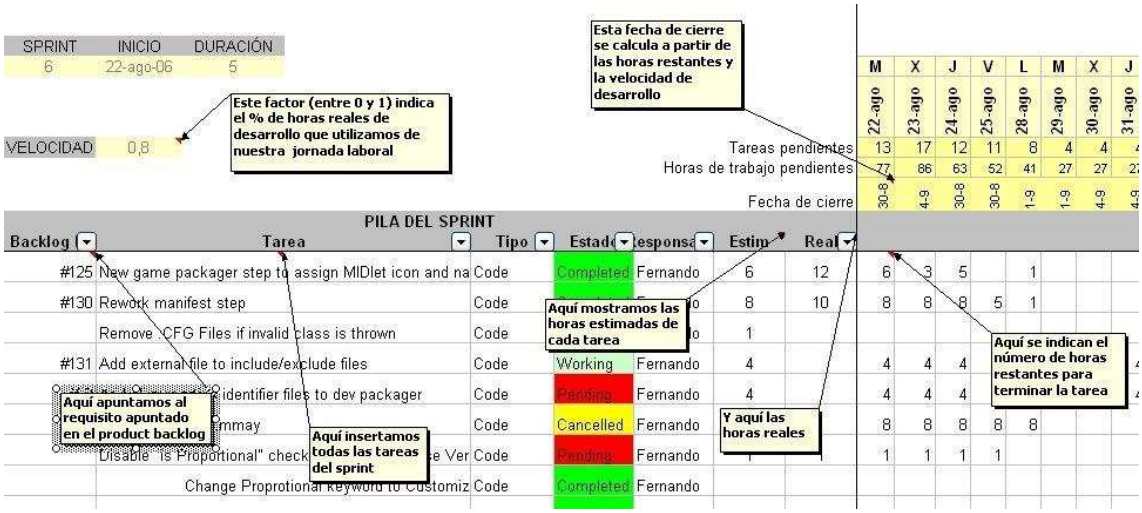

Figura 4.SCRUM. Ejemplo de Sprint Backlog (Gúzman Matus)

- Sprint Planning Meeting: (Gúzman Matus)

El Sprint Planning Meeting es una reunión que tiene por objetivo, planificar el Sprint a partir del Product Backlog. El objetivo de esta reunión es la de mover las tareas del Product Backlog al Sprint Backlog.

Del Sprint Planning Meeting, sale también el Sprint Goal, que es un pequeño documento o una breve descripción que indica lo que el Sprint intentará alcanzar.

La reunión de planeamiento del Sprint se divide en dos partes. Las primeras cuatro horas se dedican al Product Owner que presenta la prioridad más alta del Product Backlog al equipo. El Team le pregunta a él sobre el contenido, el propósito, el significado, y las intenciones del Product Backlog. Cuando el equipo sabe bastante, pero antes de que las primeras cuatro horas pasen, el equipo selecciona del Product Backlog lo que cree poder transformar en un incremento de funcionalidad para el final del Sprint. Durante las segundas cuatro horas de la reunión de planeamiento del Sprint, el equipo planea su propio Sprint. Porque el equipo es

responsable de manejar su propio trabajo, necesita un plan para comenzar con el Sprint. Las tareas que componen este plan se ponen en un Sprint Backlog; las tareas en el Sprint Backlog emergen mientras que el Sprint se desarrolla. Al comienzo del segundo período de cuatro horas en la reunión de planeamiento del Sprint, comienza a correr el tiempo, y el reloj avanza rápidamente hacia el límite del Sprint (30 días).

- Daily Scrum Meeting: (Gúzman Matus)

El Daily Scrum Meeting es una tarea iterativa que se realiza todos los días que dure el Sprint Backlog con el equipo de desarrollo o de trabajo. Se trata de una reunión operativa, informal y ágil -se realiza de pie para mantener el máximo de concentración y atención- de un máximo de 30 minutos, en la que se le hace 3 preguntas a cada integrante del equipo.

1. Qué tareas ha realizado desde la última reunión (*qué he hecho*).
2. Sobre qué va a trabajar en el día actual (*qué voy a hacer hoy*).
3. Identificación de obstáculos o riesgos que impiden o pueden impedir el normal avance (*qué ayuda necesito*). El Scrum Master, debe eliminar aquí cualquier obstáculo que encuentre.

El propósito de la reunión es sincronizar el trabajo de todos los miembros del Team y programar cualquier reunión que el Team necesite para seguir avanzando.

La reunión se hace siempre a una hora predefinida, normalmente por la mañana. Es importante que todos los miembros del equipo acudan puntuales. Todos los roles son bienvenidos, pero sólo los comprometidos pueden hablar. Uno de los puntos más importantes es el de la transparencia: todos los miembros saben que están haciendo los demás, y los problemas deben ser sacados a la luz en cuanto se detectan.

- Sprint Review: (Gúzman Matus)

Al final del Sprint, se realiza una reunión de revisión de Sprint. Ésta es una reunión de unas cuatro horas, es la reunión tiempo-limitada en la cual el Team presenta qué fue desarrollado durante el Sprint al Product Owner y a los otros interesados. Esta reunión informal, en la cual se presenta la funcionalidad, se hace para reunir a todos los participantes y que colaboren con ideas de lo que se debería hacer a continuación.

- Sprint Retrospective: (Gúzman Matus)

Después del Sprint Review y antes de la reunión de planeamiento del Sprint, el ScrumMaster convoca a una Sprint Retrospective del Sprint con el equipo. En esta reunión de tres horas, tiempo-limitada, el ScrumMaster hace que el Team revise, dentro del marco y de las prácticas de proceso, su proceso de desarrollo SCRUM, para hacerlo más eficaz y agradable para el próximo Sprint. En conjunto, Sprint Planning Meeting, Daily Scrum, Sprint Review, y el Sprint Retrospective, constituyen la inspección empírica y prácticas de la adaptación del Scrum, como se muestra en la Figura 5.SCRUM. “Diagrama de Scrum”.

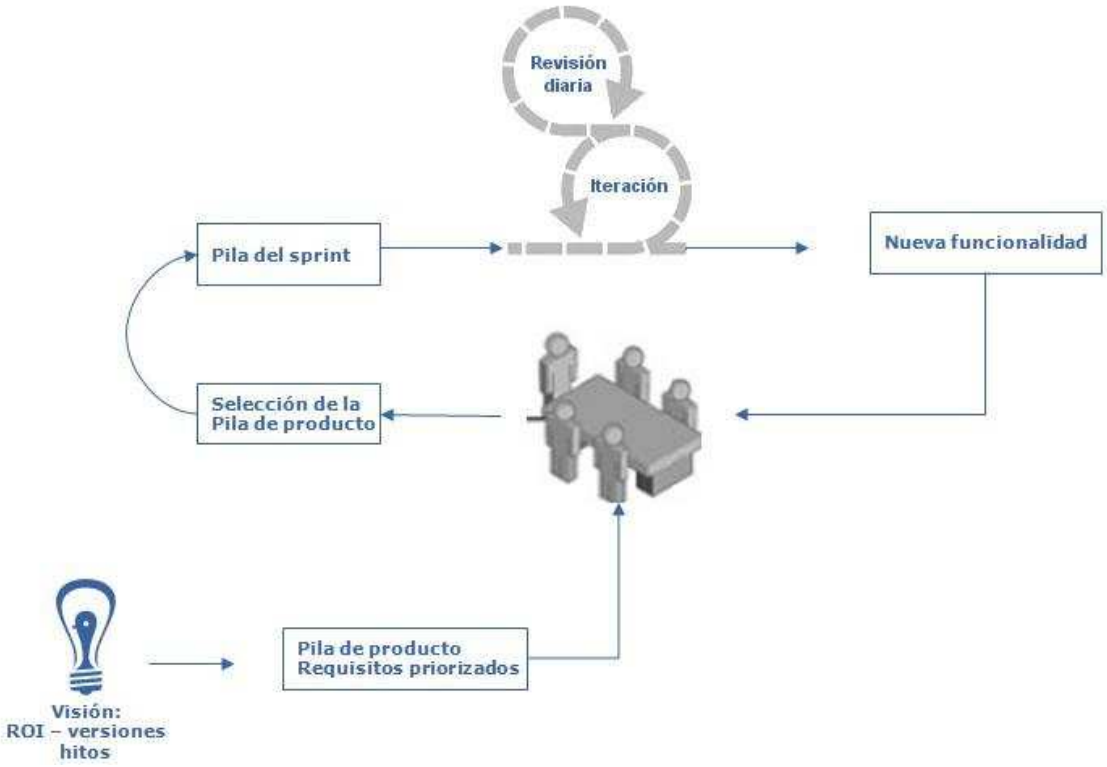

Figura 5.SCRUM. Diagrama de Scrum (Gúzman Matus)

- Carta Burndown: (Gúzman Matus)

Una carta del burndown demuestra la cantidad de trabajo restante a través de tiempo. La carta burndown es una manera excelente de visualizar la correlación entre la cantidad de trabajo restante en cualquier punto y el progreso de los equipos de proyecto en la reducción de este trabajo. La intersección de una línea de la tendencia para el trabajo restante y el eje horizontal indica el punto más probable en el que terminen las actividades. Un ejemplo de carta burndown que refleja esto se muestra en la Figura 6.SCRUM. "Ejemplo de Carta Burndown". La carta del burndown es la colisión de la realidad (trabajo hecho y cuán rápidamente se está haciendo) con lo planeado, o lo que se espera.

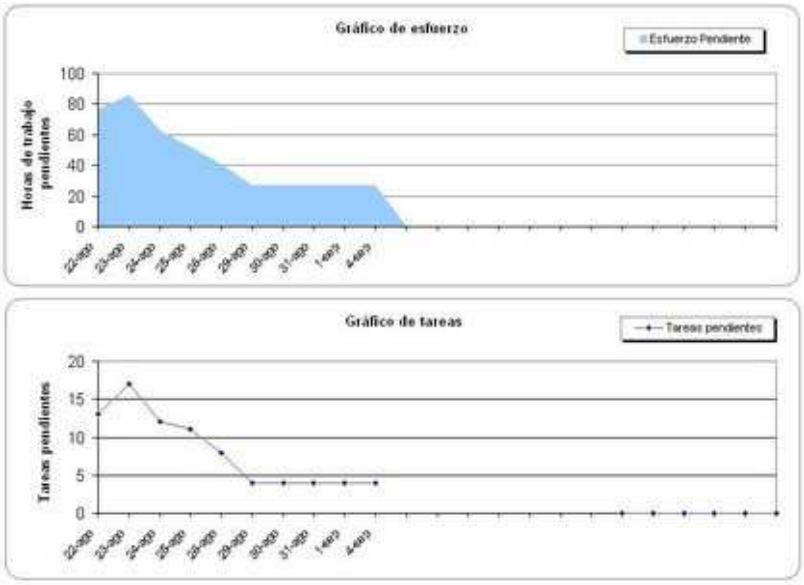

Figura 6.SCRUM. Ejemplo de Carta Burndown (Gúzman Matus)

2.2. RUP (Rational Unified Process)

Los orígenes de RUP se remontan al modelo espiral original de Barry Boehm. Ken Hartman, uno de los contribuidores claves de RUP colaboró con Boehm en la investigación. En 1995 Rational Software compró una compañía sueca llamada Objectory AB, fundada por Ivar Jacobson, famoso por haber incorporado los casos de uso a los métodos de desarrollo orientados a objetos. El Rational Unified Process fue el resultado de una convergencia de Rational Approach y Objectory (el proceso de la empresa Objectory AB). El primer resultado de esta fusión fue el Rational Objectory Process, la primera versión de RUP, fue puesta en el mercado en 1998, siendo el arquitecto en jefe Philippe Kruchten. (Wikipedia, 2008)

El Proceso Unificado de Rational (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (Wikipedia, 2008)

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. (Wikipedia, 2008)

RUP define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. Como características esenciales está dirigido por los casos de uso: que orientan el proyecto a la importancia para el usuario y lo que este quiere, está centrado en la arquitectura: que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y es iterativo e incremental: donde divide el proyecto en miniproyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada. (Gómez Gallego, 2007)

Como filosofía RUP maneja 6 principios clave: (Gómez Gallego, 2007)

- *Adaptación del proceso*: El proceso deberá adaptarse a las características propias de la organización.

- *Balancear prioridades*: Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados.

- *Colaboración entre equipos*: El desarrollo de software no lo hace una única persona sino múltiples equipos bien comunicados.

- *Demostrar valor iterativamente*: Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

- *Elevar el nivel de abstracción*: Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. (Wikipedia, 2008)

- *Enfocarse en la calidad*: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. (Wikipedia, 2008)

2.2.1. Ciclo de Vida de RUP (Wikipedia, 2008) (Gómez Gallego, 2007)

El ciclo de vida RUP es una implementación del "Desarrollo en Espiral". Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. En la Figura RUP1."Ciclo de vida de RUP" se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

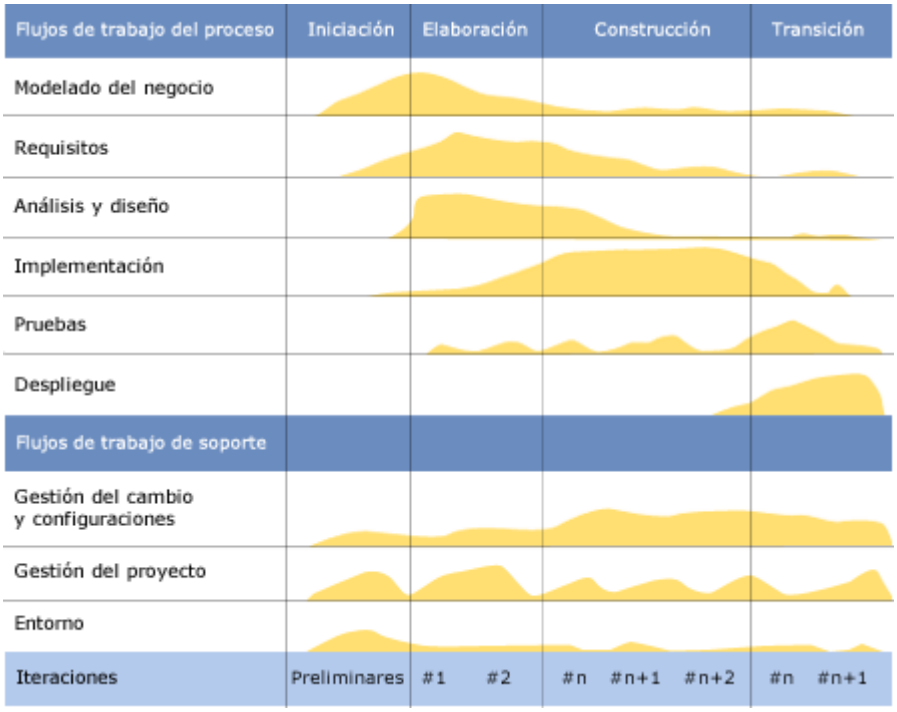

Figura 1.RUP. Ciclo de vida de RUP (Wikipedia, 2008)

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una baseline de la arquitectura.

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos. Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la baseline de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la baseline de la arquitectura, se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones, se concentra en la elaboración de un producto totalmente operativo y eficiente y del manual de usuario.

Para cada iteración se seleccionan algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios, se instala el producto en el cliente y se entrenan a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

2.2.2. Principales Características de RUP (Wikipedia, 2008)

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo (reduce riesgos).
- Administración de requisitos.
- Uso de arquitectura basada en componentes (nuevos o existentes).
- Control de cambios.
- Modelado visual del software (esto lo logra mediante el uso de UML).
- Verificación de la calidad del software.

El RUP es un producto de Rational (IBM) que incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

2.2.3. Fases de RUP (Wikipedia, 2008)

RUP, a nivel de fases, contiene una estructura estática (actividades, artefactos, flujos de trabajo) y una estructura dinámica (ciclos, iteraciones, fases, hitos). La estructura estática de RUP comprende dos aspectos importantes por los cuales se establecen las disciplinas:

Proceso: Las etapas de esta sección son modelado de negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue.

Soporte: En esta parte nos conseguimos con las siguientes etapas: gestión del cambio y configuraciones, gestión del proyecto y entorno.

La estructura dinámica de RUP es la que permite que este sea un proceso de desarrollo fundamentalmente iterativo, y en esta parte se ven inmersas las cuatro fases descritas anteriormente:

- Inicio (También llamado Incepción).
- Elaboración.
- Desarrollo (También llamado Implementación o Construcción).
- Cierre (También llamado Transición).

2.2.4. Artefactos de RUP (Wikipedia, 2008)

RUP en cada una de sus fases (pertenecientes a la estructura estática) realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema (entre otros). Estos artefactos (entre otros) son los siguientes:

a. Inicio:

- Especificación de Requerimientos.
- Visión del producto y su alcance.

- Entidades externas con las que interactuará.
- Lista de Casos de Uso.
- Riesgos.
- Plan del Proyecto.

b. Elaboración:

- Características y diseño de la arquitectura.
- Eliminar los riesgos más altos.
- 80% de los casos de uso completos.
- Requerimientos no funcionales.
- Prototipo ejecutable.

c. Construcción:

- Construir el producto hasta que esté listo para salir al mercado.
- El producto se integra sobre la plataforma adecuada.
- Manuales de Usuario.
- Documento Arquitectura que trabaja con las siguientes vistas:
    - Vista Lógica:
        - Diagrama de clases.
        - Modelo E-R (Si el sistema así lo requiere).
    - Vista de Implementación:
        - Diagrama de Secuencia.
        - Diagrama de Estados.
        - Diagrama de Colaboración.
    - Vista Conceptual:
        - Modelo de dominio.
    - Vista física:
        - Mapa de comportamiento a nivel de hardware.

d. Transición:

- Transición del producto a los usuarios.
- Manufactura, envío, soporte y mantenimiento.
- Ajustes, incluyendo corrección de errores.

2.2.5. Actividades de RUP (Gómez Gallego, 2007)

Las actividades a realizar dentro de cada una de las fases son:

a. Fase de Inicio:

Durante la fase de inicio las iteraciones ponen mayor énfasis en actividades de modelado del negocio y de requisitos.

- Modelado del negocio: En esta fase el equipo se familiarizará más al funcionamiento de la empresa, sobre conocer sus procesos.

- Requisitos: En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se

especifican. Definir el ámbito del sistema así como una interfaz de usuarios enfocada a las necesidades y metas del usuario.

b. Fase de Elaboración:

En la fase de elaboración, las iteraciones se orientan al desarrollo de la baseline de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la baseline de la arquitectura.

- Análisis y Diseño: En esta actividad se especifican los requerimientos y se describen sobre cómo se van a implementar en el sistema.

c. Fase de Construcción:

- Implementación: Se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. El resultado final es un sistema ejecutable. Planificar qué subsistemas deben ser implementados y en qué orden deben ser integrados, formando el Plan de Integración.

- Pruebas: Este flujo de trabajo es el encargado de evaluar la calidad del producto que se está desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

- Despliegue: Esta actividad tiene como objetivo producir con éxito distribuciones del producto y distribuirlo a los usuarios.

d. Durante todo el proyecto:

- Gestión del proyecto: Se vigila el cumplimiento de los objetivos, gestión de riesgos y restricciones para desarrollar un producto que sea acorde a los requisitos de los clientes y los usuarios.

- Configuración y control de cambios: El control de cambios permite mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido.

- Entorno: La finalidad de esta actividad es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, así como definir la instancia concreta del proceso que se va a seguir.

2.2.6. Roles en RUP (Gómez Gallego, 2007)

Los roles en RUP son:

Analistas:
- Analista de procesos de negocio.
- Diseñador del negocio.

Especialista en pruebas:
- Especialista en Pruebas (tester).
- Analista de pruebas.

- Analista de sistema.
- Especificador de requisitos.
- Diseñador de pruebas.

Gestores:

- Jefe de proyecto.
- Jefe de control de cambios.
- Jefe de configuración.
- Jefe de pruebas.
- Jefe de despliegue.
- Ingeniero de procesos.
- Revisor de gestión del proyecto.
- Gestor de pruebas.

Desarrolladores:

- Arquitecto de software.
- Diseñador.
- Diseñador de interfaz de usuario
- Diseñador de cápsulas.
- Diseñador de base de datos.
- Implementador.
- Integrador.

Apoyo:

- Documentador técnico.
- Administrador de sistema.
- Especialista en herramientas.
- Desarrollador de cursos.
- Artista gráfico.

Otros roles:

- Stakeholders[1].
- Revisor.
- Coordinación de revisiones.
- Revisor técnico.
- Cualquier rol.

Notas

Para grandes organizaciones con un gran número de equipos de ingenieros, la comunicación entre cada equipo es crítica por lo tanto es necesario que los artefactos sean completos y bastante comprensivos. En tanto que para pequeños proyectos no es recomendable presentarse tanto rigor en las preparaciones de los artefactos, la eficiencia del proceso depende más de las habilidades de cada trabajador. (Gómez Gallego, 2007)

La metodología RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios. (Wikipedia, 2008)

[1] Stakeholders: El término fue utilizado por primera vez por R. E. Freeman en su obra: "*Strategic Management: A Stakeholder Approach*", (Pitman, 1984) para referirse a quienes pueden afectar o son afectados por las actividades de una empresa.

CAPÍTULO 3: MARCO APLICATIVO

En el presente capítulo se describe de forma detallada el desarrollo de la aplicación de programa de incentivos desde su concepción hasta su puesta en marcha. Se plantea el problema para definir el objetivo general y los objetivos específicos, delimitando el alcance del proyecto para luego describir el ambiente de ejecución del sistema y la plataforma tecnológica usada. Y por último se desglosa la metodología de desarrollo de software que se siguió durante el desarrollo del proyecto.

3.1. Planteamiento del Problema

Se requiere automatizar los procesos dentro de un programa de incentivos a fin de disminuir los costos para las empresas referentes a este rubro y conseguir así un verdadero aumento en la productividad de la empresa, por lo tanto, se desarrolló una aplicación web que sirve de base para la realización de programas de incentivos personalizados para diferentes empresas mediante la fácil modificación de la misma de acuerdo a las necesidades e imagen de cada empresa.

El programa de incentivos está basado en la imagen de "club" que tiene participantes (clientes y/o empleados).

En este sistema dichos participantes se afilian (esto permite a la empresa conocer quiénes le compran y/o venden sus productos), pudiendo luego registrar las diferentes constancias (facturas, recibos, etc.) de las compras y/o ventas que realizan (qué compran los clientes finales) indicando la fecha de la misma (cuándo compran).

Con el programa de incentivos se persiguen los siguientes objetivos generales:

- Incrementar las ventas de los productos de la empresa.
- Aumentar el conocimiento de la marca en sus sucursales y personal.
- Reconocer y motivar a los clientes y/o empleados de la empresa.
- Utilizar canales de comunicación más eficientes y efectivos.

El programa de incentivos requiere un desarrollo en cuatro grandes aspectos: afiliación, generación de puntos, redención o canje de recompensas y administración.

La afiliación al programa de incentivos está dirigida a los potenciales participantes (empleados o clientes) y el proceso abarca, de manera general, la obtención de los principales datos del usuario, la aprobación y generación de puntos por afiliación (en caso de aplicarse) y la notificación.

La generación de puntos dentro del club puede darse por la compra y/o venta de los productos de la empresa, por la cantidad y/o monto de facturas ingresadas al sistema o por la afiliación al club. La empresa decide cuáles de estas reglas activar durante la configuración.

El proceso de canjear puntos por recompensas implica el despliegue de un catálogo donde el participante podrá seleccionar, según la cantidad de puntos que posee, los productos que desea obtener.

La administración del club abarca los perfiles de los usuarios del sistema, las sucursales del club y la visualización de reportes.

3.2. Objetivo General

Construir una aplicación web que sirva como base para el desarrollo de programas de incentivos, a través de la fácil modificación de la interfaz según la imagen de la empresa así como las reglas de negocio que deben aplicar, utilizando una metodología de desarrollo híbrida entre RUP y Scrum, donde se tomen los aspectos de cada una que permitan desarrollar un buen producto con tiempos de espera cortos.

3.3. Objetivos Específicos

- Analizar y diagramar los procesos y procedimientos de un programa de incentivos, características, ventajas y desventajas, que permita conocer con alto nivel de detalle la forma de ejecución de cada uno de ellos y sus posibles flujos, para determinar de una manera clara y específica los requerimientos de este.

- Proponer un plan de desarrollo basado en métodos, herramientas y experiencias previas de usuarios, que negociado con el cliente permita definir una arquitectura base para la construcción del sistema.

- Prever posibles riesgos en cada una de las diferentes fases de desarrollo y tener planes alternos que permitan una solución rápida y oportuna.

- Diseño Físico y Lógico de la Base de Datos que permita almacenar la información relacionada a la problemática planteada.

- Desarrollar un módulo para la administración del programa de incentivos (club), el cual le permita al cliente administrar las cuentas de usuario, las sucursales, los productos participantes y visualizar reportes.

- Desarrollar un módulo para la administración de la tienda, el cual le permita al administrador de mercadeo administrar el catálogo de recompensas, las órdenes de despacho y visualizar reportes.

- Desarrollar un módulo para la participación en el programa de incentivos, el cual le permita a los usuarios no registrados afiliarse al club y a los participantes activos del club registrar las ventas, canjear los puntos obtenidos por recompensas y visualizar reportes individuales.

- Realizar las pruebas de la herramienta desarrollada para verificar el correcto funcionamiento de la misma.

3.4. Límites y Alcance

La aplicación Web debe estar capacitada para:

- Servir de base para la creación de un programa de incentivos modificando la interfaz del mismo (logo, colores, textos, etc.) y las reglas de negocio que van a ser aplicadas en dicho programa de incentivos.

- Permitir, dentro del programa de incentivos, la afiliación de los participantes, el registro de las compras y/o ventas realizadas, el canje de los puntos (o el nombre dado por la empresa a la moneda utilizada en el club) por recompensas, la administración de su cuenta, la visualización de reportes referentes: a las compras y/o ventas realizadas por él y por los participantes a su cargo (de cargo inferior en la pirámide organizacional de la empresa), a los canjes realizados por él, y a su estado de cuenta histórico y actual.

- Permitir, dentro del programa de incentivos, la administración de las cuentas de los participantes, la administración de la reglas de negocio, la generación de invitaciones para afiliación, la administración de sucursales, la visualización de reportes referentes: a las compras y/o ventas realizadas por las sucursales, a los canjes realizados por las sucursales, y a los rankings de sucursales y participantes.

- Permitir la administración de la tienda (catálogo de productos, órdenes de compra y órdenes de despacho), así como la visualización de los reportes referentes a los canjes que son realizados en el programa de incentivos.

3.5. Justificación e Importancia

El desarrollo de este Trabajo Especial de Grado permitirá la obtención de los siguientes beneficios:

- Generar un mejoramiento en la productividad, integración de procesos y simplificación de actividades, que se traduce en un incremento de la eficiencia en el cumplimiento de las actividades relacionadas con el mercadeo de los productos de la empresa.

- Optimizar el proceso de generación de programas de incentivos logrando así minimizar los tiempos de creación de la aplicación web final.

- Crear un medio de colaboración e interacción entre los clientes finales, distribuidores, personal y gerentes de mercadeo de las empresas que pongan en marcha un programa de incentivos.

- Brindar a los usuarios la facilidad de acceso a toda la información de las ventas y/o compras de los productos de la empresa de una manera rápida, confiable, cómoda y eficiente, vía Web.

- Permitir la manipulación más consistente de toda la información con el manejo de perfiles de usuarios.

3.6. Ambiente de Ejecución

Las características del ambiente en el cual fue instalada la aplicación web son:

- Cliente: Marketing Solutions
- Ubicación: Vsdaycoesx03
- Host Name: VMARKETING01
- CPU: Dual 3.00 GHz Intel Zeon
- IP Address: 200.74.222.195, 200.74.222.196, 172.28.76.43
- Default Gateway: 200.74.222.193
- DNS (Server): (none) 200.74.222.140, 200.74.222.148 (none)
- Free Space: C:\ 3.63 GB NTFS, E:\ 47.89 GB NTFS, F:\ 69.62 GB NTFS
- Volumes: C:\ 19.99 GB NTFS, E:\ 49.99 GB NTFS, F:\ 69.99 GB NTFS
- MAC Address: 00-50-56-81-0D-6F, 00-50-56-81-26-BA, 00-50-56-B2-7C-3E
- Memory: 2048 MB
- OS Version: Windows 2003
- Service Pack: Service Pack 2
- Subnet Mask: 255.255.255.248, 255.255.255.248, 255.255.255.0
- System Type: Server, Stand-alone, Terminal Server
- User Name: admmark

3.7. Plataforma Tecnológica Utilizada

Las tecnologías a utilizar para la elaboración de la aplicación de Programas de Incentivos son SQL Server 2005 como Servidor de Base de Datos y IIS (Internet Information Server) como Servidor Web. La razón de la elección de estos es porque el primer cliente del producto de software así lo solicitó. Además ambos cuentan con una serie de características que cubren las necesidades del sistema a implementar.

SQL Server es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la base de datos SQL Server ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento para utilizar en su negocio.

La integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business Intelligence Development Studio, distingue a SQL Server.

SQL Server 2005 está creado para reducir el tiempo muerto tanto planeado como inesperado, provee de soluciones para recuperación de desastres y provee de mayor disponibilidad del sistema a usuarios de la base de datos a través de tecnologías de alta disponibilidad.

Por su parte, el IIS es un componente de Windows incluido en las versiones profesionales de Windows 2000 y XP. Éste viene de forma gratuita con el sistema operativo Windows NT, 2000 y XP y descargable para los demás. El acceso al IIS se realiza mediante el icono de “Servicios de Internet Information Server” situado en las “Herramientas administrativas” dentro del panel de control. El servidor Web IIS permite administrar las aplicaciones Web y comunicarse con los navegadores cliente mediante el protocolo HTTP/HTTPS (protocolo de transferencia de

hipertexto). El IIS también ofrece otros servicios de protocolo, como transferencia de archivos (FTP), servicio de correo electrónico (SMTP) y servicio de noticias (NNTP).

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

La tecnología a emplear para la implementación de este software es ASP .NET, ya que permite un desarrollo fácil y rápido de aplicaciones Web, esto porque posee herramientas superiores de desarrollo (edición WYSIWYG, la creación de controles mediante "drag-and-drop", etc.) y funciones que abstraen casi toda la complejidad de trabajar con bases de datos, definir clases e instanciar objetos. Ofrece flexibilidad, simplicidad, escalabilidad, disponibilidad, personalización, extensibilidad y seguridad. Además la plataforma Microsoft .NET ofrece actualmente compatibilidad integrada para lenguajes como: C#, C++, Visual Basic, J# y JScript.

3.8. Metodología de Desarrollo de Software Utilizada

La metodología utilizada para el desarrollo de este Trabajo Especial de Grado fue un híbrido entre la metodología Scrum y la metodología RUP. Se procuró seleccionar de cada una de ellas los elementos que mejor se acoplarán al proyecto y las mejores prácticas que añadieran valor al producto final disminuyendo los tiempos de entrega, por lo que de la metodología RUP se adoptó el formalismo en cuanto a la documentación y de la metodología Scrum la rapidez para la implementación de las funcionalidades del sistema. Tomando en cuenta esto se dividió el trabajo en fases según RUP, es decir, se enmarcó el proyecto dentro de cuatro (4) fases (a saber): inicio, elaboración, construcción y transición. Para la fase de inicio, elaboración y transición se siguieron las buenas prácticas que identifican a la metodología RUP y para la fase de construcción las que identifican a la metodología Scrum.

En la fase de inicio se analizó el problema y se determinaron las principales funcionalidades del sistema, y se realizaron las primeras versiones del modelo conceptual y modelo de datos e interfaz del sistema tomando en cuenta la navegabilidad dentro de la aplicación web. En la fase de elaboración se refino el análisis y se determinaron el resto de las funcionalidades del sistema. La fase de construcción se desarrolló de acuerdo a la metodología Scrum realizando un listado de las funcionalidades del sistema agrupadas por módulos, según su afinidad, e implementándolas semanalmente de modo que al final de cada semana se obtuvo un incremento funcional de la aplicación. Finalmente en la fase de transición se instaló el producto y se realizaron pruebas colectivas mediante una matriz de pruebas.

A continuación se describen de manera general las actividades llevadas a cabo en cada fase y se detalla el conjunto de artefactos implementados.

3.8.1. Fase de Inicio

En la fase de inicio, a partir del levantamiento de información que realiza el líder de proyecto a través de reuniones con el cliente, se llevan a cabo las actividades de modelado de requerimientos y del negocio. Para la primera actividad se definen los usuarios del sistema y el alcance del proyecto, para la segunda actividad se familiariza con el funcionamiento de los programas de incentivos para identificar así los principales casos de uso. Esta fase se extiende por quince (15) días.

A continuación se describen los usuarios del sistema: usuario no registrado, participante, administrador de mercadeo y cliente, partiendo del diagrama de casos de uso de nivel 0.

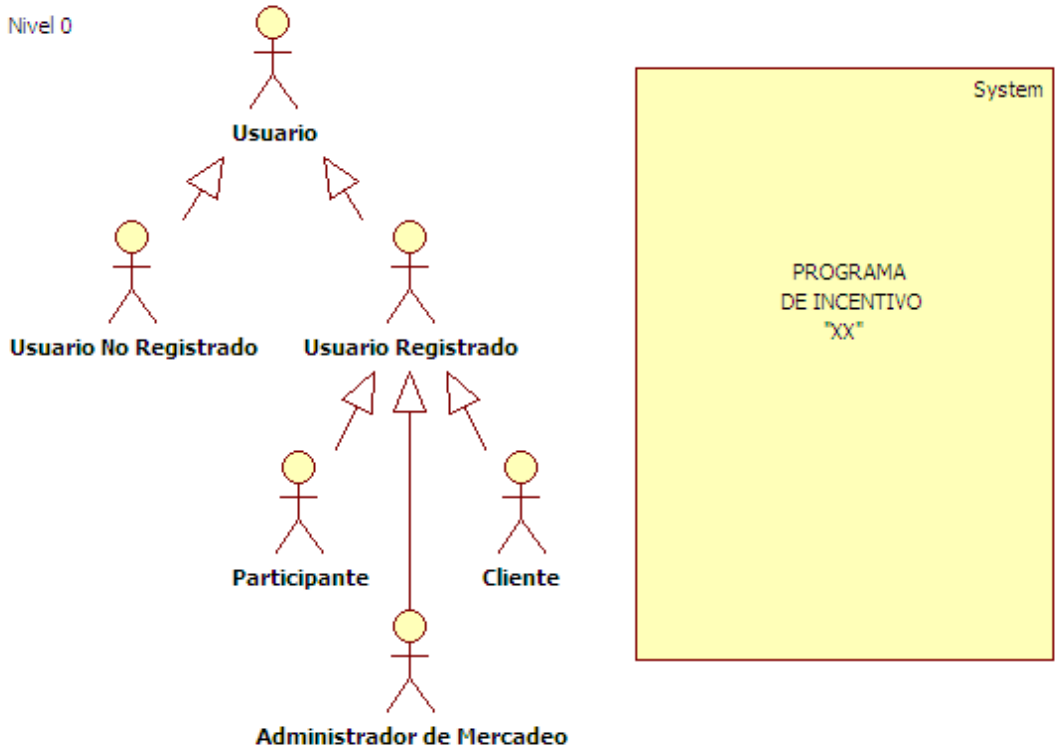

Figura 1.MDSU. Diagrama de casos de uso nivel 0

Nombre	Usuario demo (no registrado) ☺
Problema	El usuario desea conocer el sitio web sin comprometerse a nada.
Solución	El programa de incentivos le permite a cualquier persona navegar el mismo sin tener que autenticarse como usuario registrado y sin poder realizar ninguno de los procesos activos en el sitio web como por ejemplo llenar un carrito de compras y confirmarlo.
Contexto	Uso de programas de lealtad o de incentivos.
Fuerzas	El usuario debe tener una conexión a Internet.

Nombre	Participante(registrado) ☺
Problema	El usuario desea hacer uso del programa de incentivos a fin de obtener recompensas. Es el usuario principal del club de lealtad, quien va a adquirir puntos y después canjearlos por recompensas.
Solución	El programa de incentivos le permite a cualquier usuario acreditarse en el mismo mediante un identificador y una clave de acceso, obtenidos con previo registro en el programa de lealtad. Un usuario participante tiene asignada una cuenta propia que mantiene información personalizada del usuario en el servidor, como por ejemplo la dirección de correo electrónico.
Contexto	Uso de programas de lealtad o de incentivos.
Fuerzas	El usuario debe tener una conexión a Internet.

Nombre	Administrador de mercadeo (registrado) ☺
Problema	El usuario desea manejar toda la logística de distribución de las recompensas y administrar el catálogo de recompensas ofrecido en el programa de incentivos.
Solución	La aplicación recoge todos los canjes realizados por los participantes del programa de incentivos y los muestra en forma de reporte al administrador permitiéndole actualizar la información de los despachos así como la de las recompensas ofrecidas en el programa de lealtad.
Contexto	Uso de programas de lealtad o de incentivos.
Fuerzas	El usuario debe tener una conexión a Internet.

Nombre	Administrador del programa de lealtad o cliente (registrado) ☺
Problema	El usuario desea administrar el programa de incentivos.
Solución	La aplicación permite administrar la información de los participantes del programa de lealtad y ofrece a este usuario acceso a los reportes sobre los canjes realizados y los puntos generados por cada participante.
Contexto	Uso de programas de lealtad o de incentivos.
Fuerzas	El usuario debe tener una conexión a Internet.

En esta primera fase se obtienen las versiones iniciales de los casos de uso por lo que se presenta a continuación el diagrama de casos de uso de nivel 1 con sus respectivas descripciones formales.

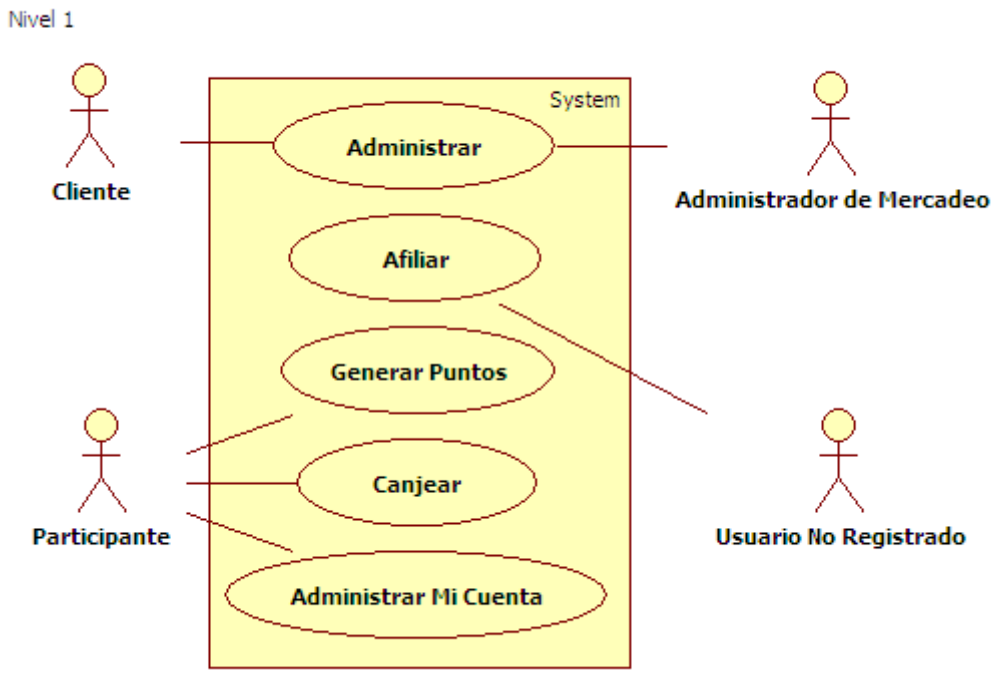

Figura 2.MDSU. Diagrama de casos de uso nivel 1

Descripción de los casos de uso

Nivel 1

Nombre: **Administrar**.
Actor: Administrador del club (cliente) y administrador de mercadeo.
Pre-Condición: El usuario ha iniciado sesión.
Flujo Principal:

1. El sistema le permite al usuario administrar el club así como las cuentas de los participantes del mismo y ver reportes.
2. El sistema le permite al usuario administrar el catálogo de recompensas así como ver las órdenes de despacho y los reportes.

Flujo alterno:
Post-Condición: Club o catalogo de recompensas administrado.

Nombre: **Afiliar**.
Actor: Usuario no registrado.
Pre-Condición: El usuario ha hecho click en el botón o enlace para registrarse.
Flujo Principal:

1. El usuario introduce sus datos de acceso de preafiliación y hace click en el botón para afiliarse al Club.
2. El sistema valida los datos ingresados:
   a. Válido:
   2.1. Se despliega el formulario de perfil de afiliación.
   b. No válido:
   2.1. Se le notifica el hecho al usuario mediante un mensaje de error en la misma página y vuelve al paso 1.
3. El usuario completa los datos de su perfil de afiliación.
4. El sistema verifica la correctitud de los datos:
   a. Correcto:
   4.1. El sistema registra el nuevo participante en la base de datos sumándole los puntos correspondientes por la afiliación.
   4.2. El sistema registra el hecho en el archivo histórico.
   4.3. El sistema envía un e-mail al administrador del club y al nuevo participante confirmándole su registro en el club y dándole la bienvenida al programa.
   4.4. El sistema notifica al usuario el éxito de la operación, finaliza y vuelve a la página de estado de cuenta del nuevo participante.
   b. No correcto:
   4.1. El sistema notifica al usuario el hecho mediante un mensaje de error, vuelve al paso 3.

Flujo Alterno: El usuario decide no completar el formulario de registro por lo que la información no es registrada.
Post-Condición: Nuevo participante afiliado al Club.

Nombre: **Generar puntos**.
Actor: Participante.
Pre-Condición: El participante ha iniciado sesión y ha hecho click en el botón para registrar una factura.
Flujo Principal:

1. El participante llena el formulario para registrar la factura.
2. El sistema verifica la completitud y correctitud de los datos ingresados:

a. Correcto:

2.1. El sistema calcula los puntos correspondientes por la compra.

2.2. El sistema registra la compra en la base de datos sumándole al participante los puntos correspondientes.

2.3. El sistema registra la generación de puntos en el archivo histórico.

2.4. El sistema notifica al usuario el éxito de la operación, finaliza y vuelve a la página de estado de cuenta del participante.

b. No correcto:

2.1. El sistema le notifica al participante el hecho mediante un mensaje de error, vuelve al paso 1.

Flujo alterno: El participante no completa el formulario por lo cual los puntos no son cargados al mismo.

Post-Condición: Puntos sumados al participante.

Nota: Tomar en cuenta que los puntos son repartidos dentro de los niveles de participantes según el porcentaje que le corresponde a cada uno.

Nombre: **Canjear**.

Actor: Participante.

Pre-Condición: El participante ha iniciado sesión y ha hecho click en el botón para canjear sus puntos.

Flujo Principal:

1. El participante selecciona el producto que desea y lo agrega al carrito de compras. Repite este paso tantas veces como desee, pudiendo también eliminarlo del carrito de compras.
2. E l participante hace click en el botón para confirmar la compra.
3. El sistema verifica que realmente pueda canjear la cantidad de puntos de que dispone por el(los) producto(s) seleccionado(s) (puntos>=∑valor_producto):

d. Valido:

1.1. El sistema registra el canje.

1.2. El sistema actualiza los puntos del participante.

1.3. El sistema registra el hecho en el archivo histórico.

1.4. El sistema envía un e-mail al participante con la información del canje, notifica el éxito de la operación y finaliza.

e. No válido:

1.1. El sistema notifica el hecho al participante mediante un mensaje de error, finaliza y vuelve al paso 1.

Flujo Alterno: El participante nunca recibió el(los) producto(s).

Post-Condición: Participante satisfecho y premiado.

Nombre: **Administrar Mi Cuenta**.

Actor: Participante.

Pre-Condición: El participante ha iniciado sesión.

Flujo Principal:

1. El sistema le permite al usuario administrar su cuenta así como modificar su contraseña y ver reportes.

Flujo Alterno:

Post-Condición: Cuenta administrada.

Una vez conocidos los usuarios y los principales casos de uso se comienza con el análisis para el diseño de las pantallas de la aplicación web, para lo cual se toman en cuenta preguntas como:

- ¿Quiénes son los usuarios, cuáles sus conocimientos, y qué pueden aprender?
- ¿Qué quieren o necesitan hacer los usuarios?
- ¿Cuál es la formación general de los usuarios?
- ¿Cuál es el contexto en el que el usuario está trabajando?
- ¿Qué debe dejarse a la máquina? ¿Qué al usuario?

Y se realizan las versiones iniciales de los diagramas WAE para representar así la posible navegabilidad del sitio web. Se presentan a continuación los diagramas WAE para la navegación en la página de inicio y la navegación del menú principal, el diagrama WAE correspondiente a la navegación del menú secundario se encuentra en el apartado “Anexos – Figura 21.A. Diagrama WAE Navegación Secundaria”.

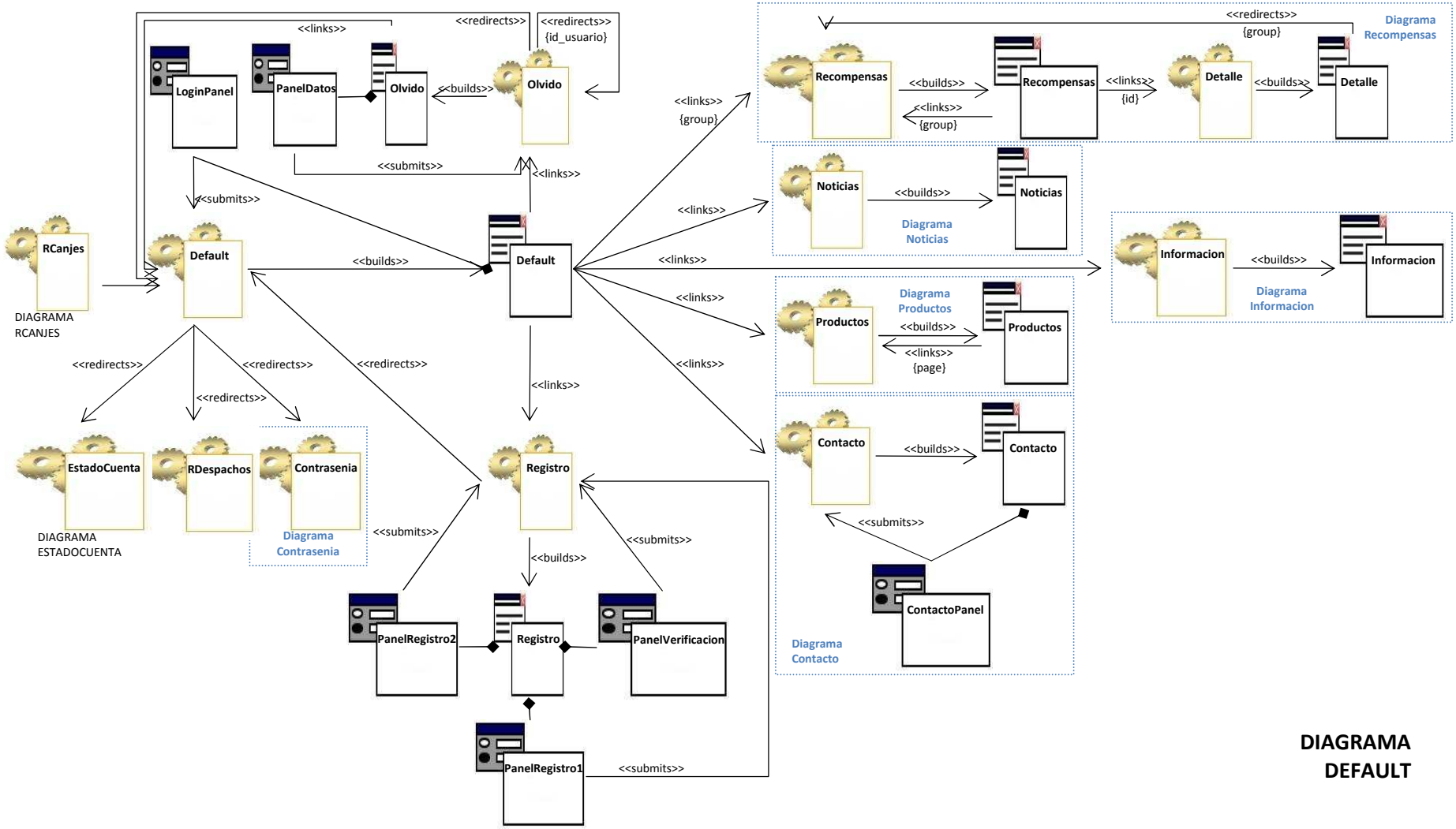

DIAGRAMA DEFAULT

Figura 3.MDSU. Diagrama WAE “Default”

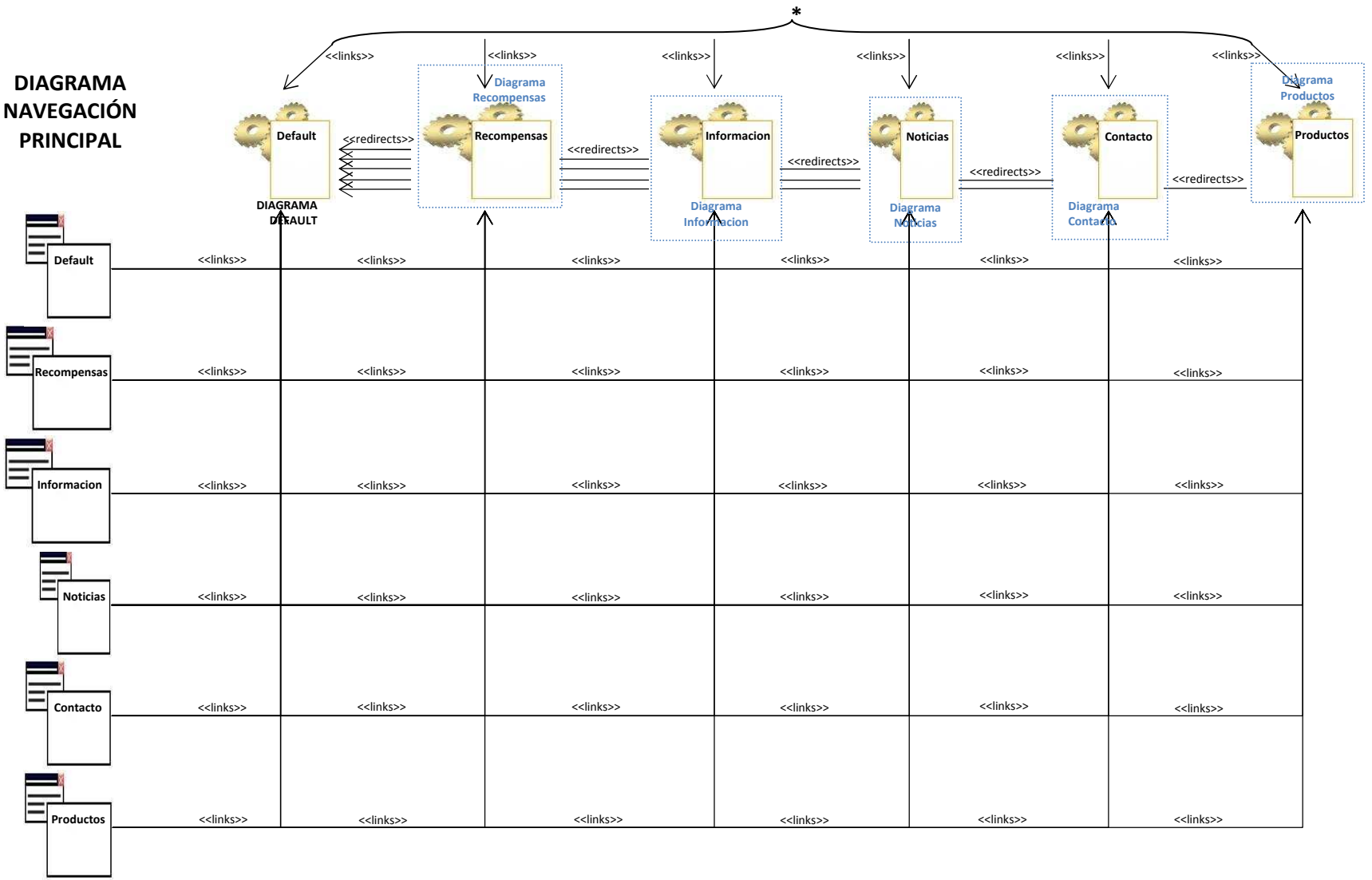

Figura 4.MDSU. Diagrama WAE “Navegación Principal”

También dentro del análisis cubierto en esta fase, y para cimentar el modelado conceptual de los programas de incentivos, se realiza el diagrama de clases, presentado a continuación.

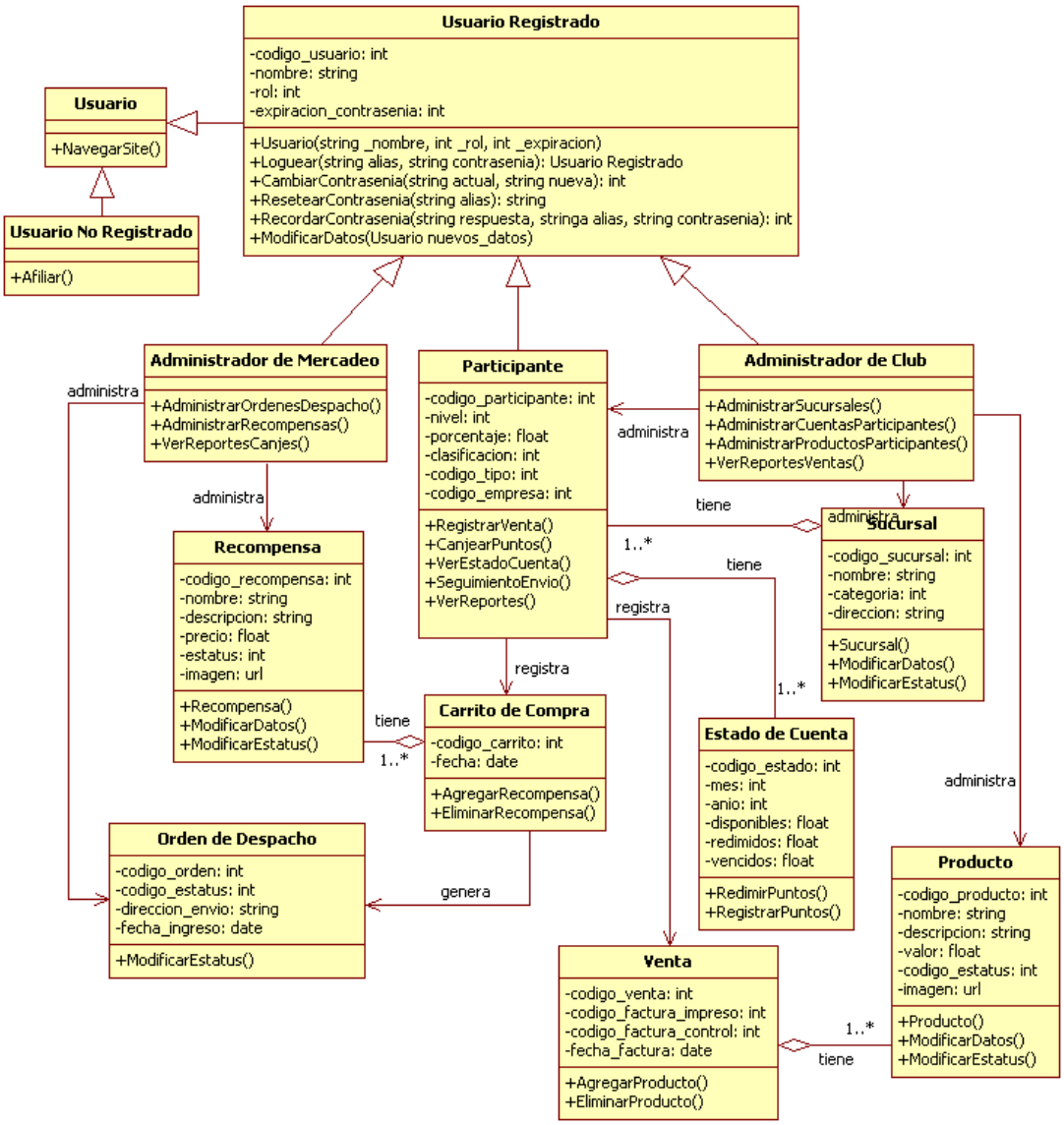

Figura 5.MDSU. Diagrama de clases

En cuanto al modelo de datos se obtienen en esta primera fase la versión inicial del diagrama entidad-relación presentado a continuación (por partes según el aspecto del programa de incentivos que cubre).

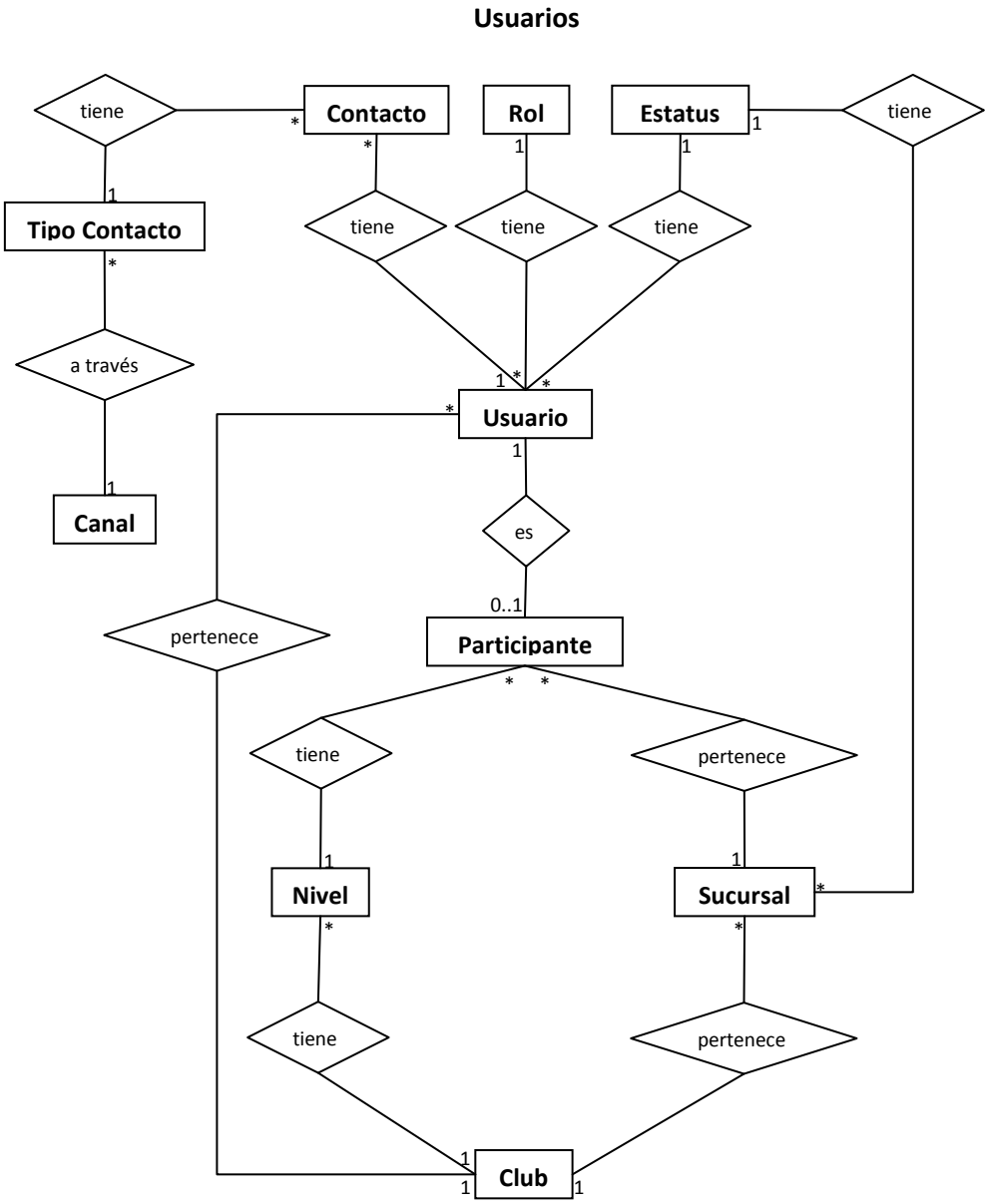

Figura 6.MDSU. Diagrama Entidad-Relación (Usuarios)

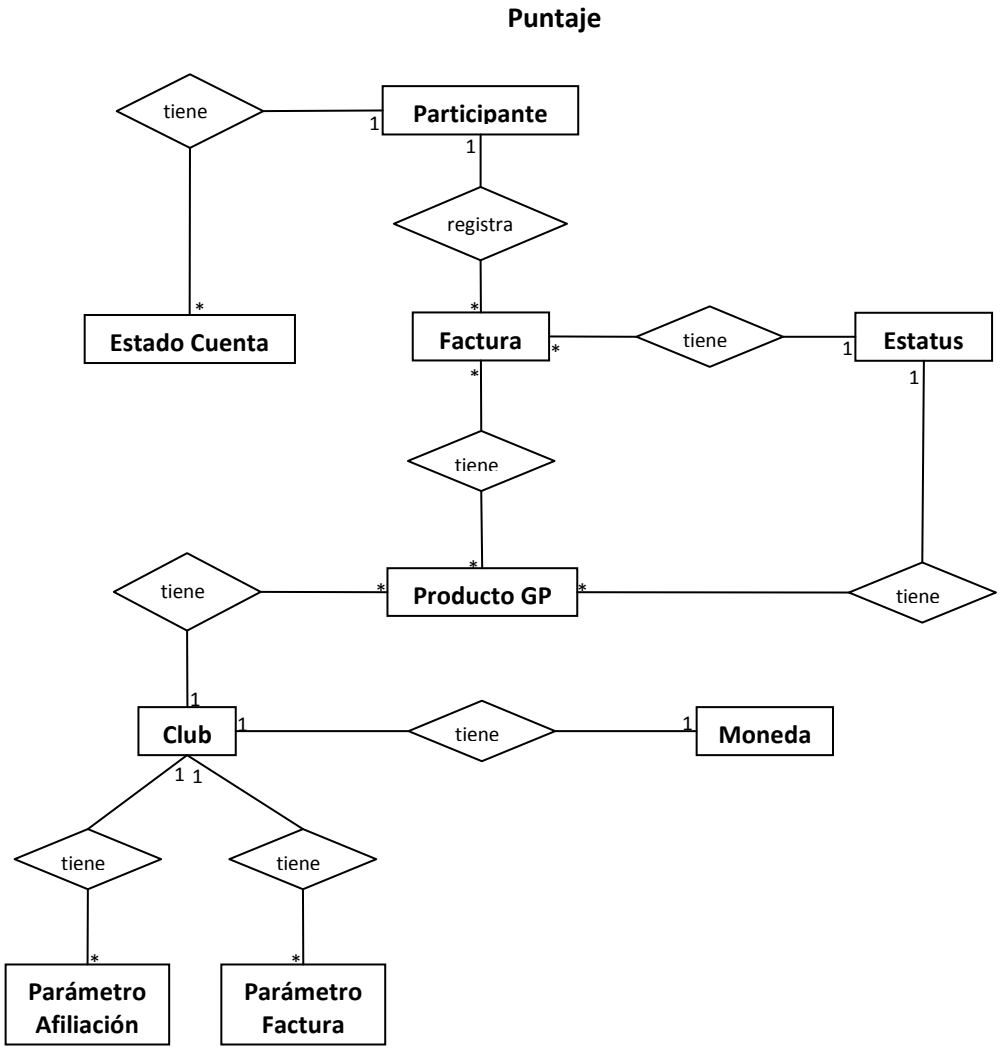

Figura 7. MDSU. Diagrama Entidad-Relación (Puntaje)

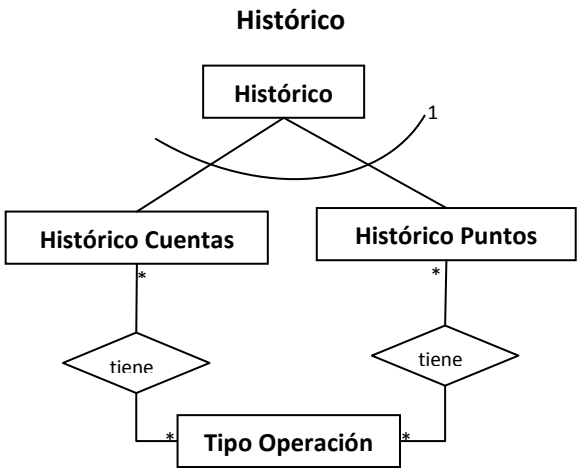

Figura 8. MDSU. Diagrama Entidad-Relación (Histórico)

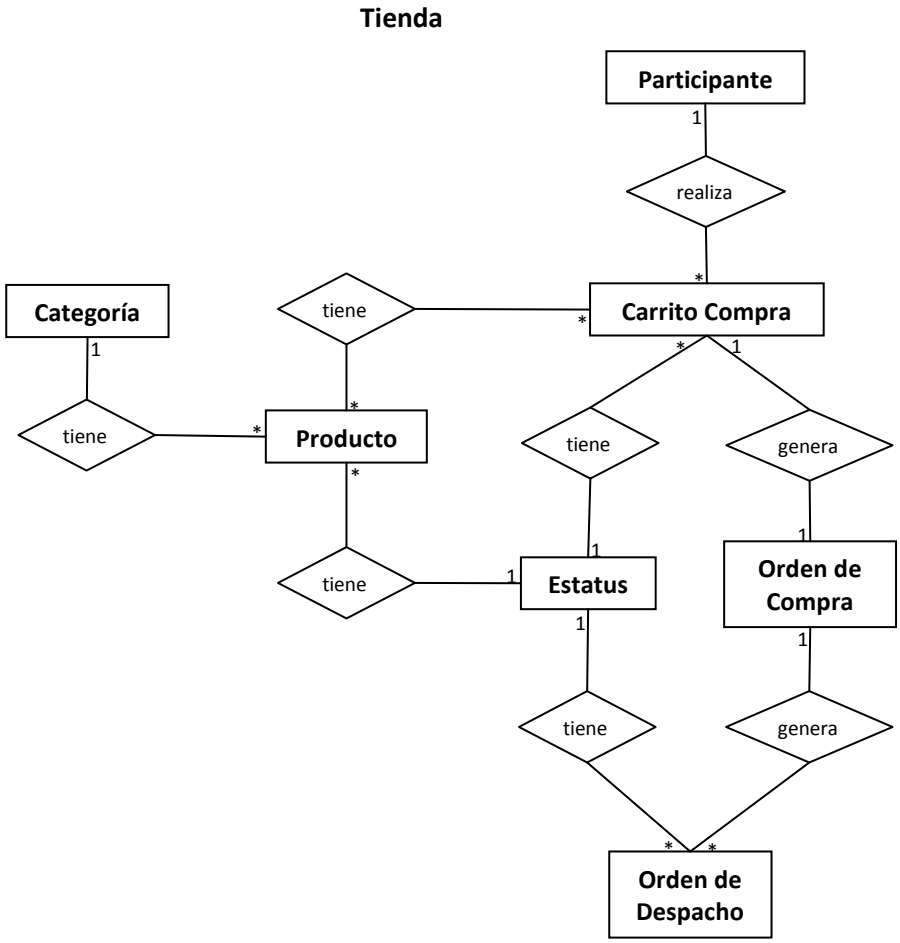

Figura 9. MDSU. Diagrama Entidad-Relación (Tienda)

3.8.2. Fase de Elaboración

La fase de elaboración, para este proyecto, abarca el refinamiento de los casos de uso y el diseño de la interfaz, y se extiende por quince (15) días.

Para el diseño de la interfaz (Figura MDSU4.” Interfaz final – Página de Inicio”) se toman en cuenta los siguientes aspectos relacionados con la usabilidad:

- Reducir al máximo el número de campos en cada formulario y no fragmentar demasiado la información, ya que cuanto más corto, sencillo y directo sea el formulario más gente lo completará correctamente;

- Para evitar la incomodidad del cambio entre teclado y ratón, es recomendable, cuando tenga sentido, agrupar por un lado los controles que se manejan con el ratón (radio-buttons, check-boxes, combos) y por otro los que se manejan con el teclado (campos de texto), en lugar de alternarlos;

- Aplicar las validaciones a nivel de formulario ya que ello permite que el usuario “navegue” libremente por los campos del formulario sin que el sistema le interrumpa

constantemente mostrándole mensajes de error por no superar una validación a nivel de campo.

- Para aplicaciones "web" es mejor usar páginas con tamaños fijos, ya que en monitores con resolución de pantalla superior a 1024 la distancia entre la información y los botones es muy grande y se pierda el contexto (botones situados en el margen derecho y el campo de texto en el margen izquierdo);

- Respecto a la situación, tanto los "radio button" como los "check-box" siempre se han de situar a la izquierda de la etiqueta del campo, así se favorece la alineación vertical de todos los controles. Por el contrario los combos y los campos de texto deben situarse a la derecha o debajo de la etiqueta del campo y se recomienda añadir un pequeño texto explicando el formato y/o rango de valores correcto;

- Usar campos con listas de valores asociadas o combo-boxes que facilitan que el usuario informe el campo con un valor correcto;

- Colocar en la barra de navegación sólo enlaces útiles u opciones fundamentales, más de cinco (5) empieza a ser demasiado, claramente definidos, no solo en su significado sino en su representación: separarlos de forma clara con una línea vertical entre unos y otros para que el usuario sepa dónde empieza uno y acaba otro;

- Es recomendable dejar los botones del menú siempre activos y procurar usar la página en sí para mostrar al usuario la información de "donde está", utilizando títulos y subtítulos y destacando los textos más importantes con negrita y subrayado para ayudar a clasificar la información, hacerla más visual y rápida para el usuario;

- Para los reportes se usarán los listados, ya que son una de las mejores interfaces de usuario, no solo por su sencillez sino por su utilidad para el usuario final. Es recomendable que los filtros y criterios de la búsqueda se integren en la propia página del listado porque esto permite visualizar más claramente el cambio, el antes y el después;

- La página principal debe orientar sobre qué trata el sitio, como está estructurado y como navegar en él. El usuario suele mirar al centro de la pantalla, por lo que se debe procurar que los elementos importantes estén así colocados;

- El uso adecuado de metáforas facilita el aprendizaje de un sitio web, pero un uso inadecuado de estas puede dificultar enormemente el aprendizaje;

Y los siguientes principios:

1. Anticipación, el sitio web debe anticiparse a las necesidades del usuario.
2. Autonomía, los usuarios deben tener el control sobre el sitio web. Los usuarios sienten que controlan un sitio web si conocen su situación en un entorno abarcable y no infinito.
3. Los colores han de utilizarse con precaución para no dificultar el acceso a los usuarios con problemas de distinción de colores (aprox. un 15% del total).
4. Consistencia, las aplicaciones deben ser consistentes con las expectativas de los usuarios, es decir, con su aprendizaje previo.

5. Eficiencia del usuario, los sitios web se deben centrar en la productividad del usuario, no en la del propio sitio web.
6. Reversibilidad, un sitio web ha de permitir deshacer las acciones realizadas.
7. Aprendizaje, los sitios web deben requerir un mínimo proceso de aprendizaje y deben poder ser utilizados desde el primer momento.
8. Interfaz visible, se deben evitar elementos invisibles de navegación que han de ser inferidos por los usuarios, menús desplegables, indicaciones ocultas, etc.

Figura 10.MDSU.Interfaz final – Página de Inicio

El diseño de la interfaz se complementa con la realización de los diagramas WAE para la navegabilidad de las páginas por usuario, a continuación se muestra la navegación para el participante por ser el usuario principal del sistema; el resto de los diagramas WAE para las páginas del cliente y del administrador de mercadeo se encuentran en el apartado “Anexos - Figura 22.A. Diagrama WAE Reporte de Canjes y Figura 23.A. Diagrama WAE Reporte de Órdenes de Despacho”.

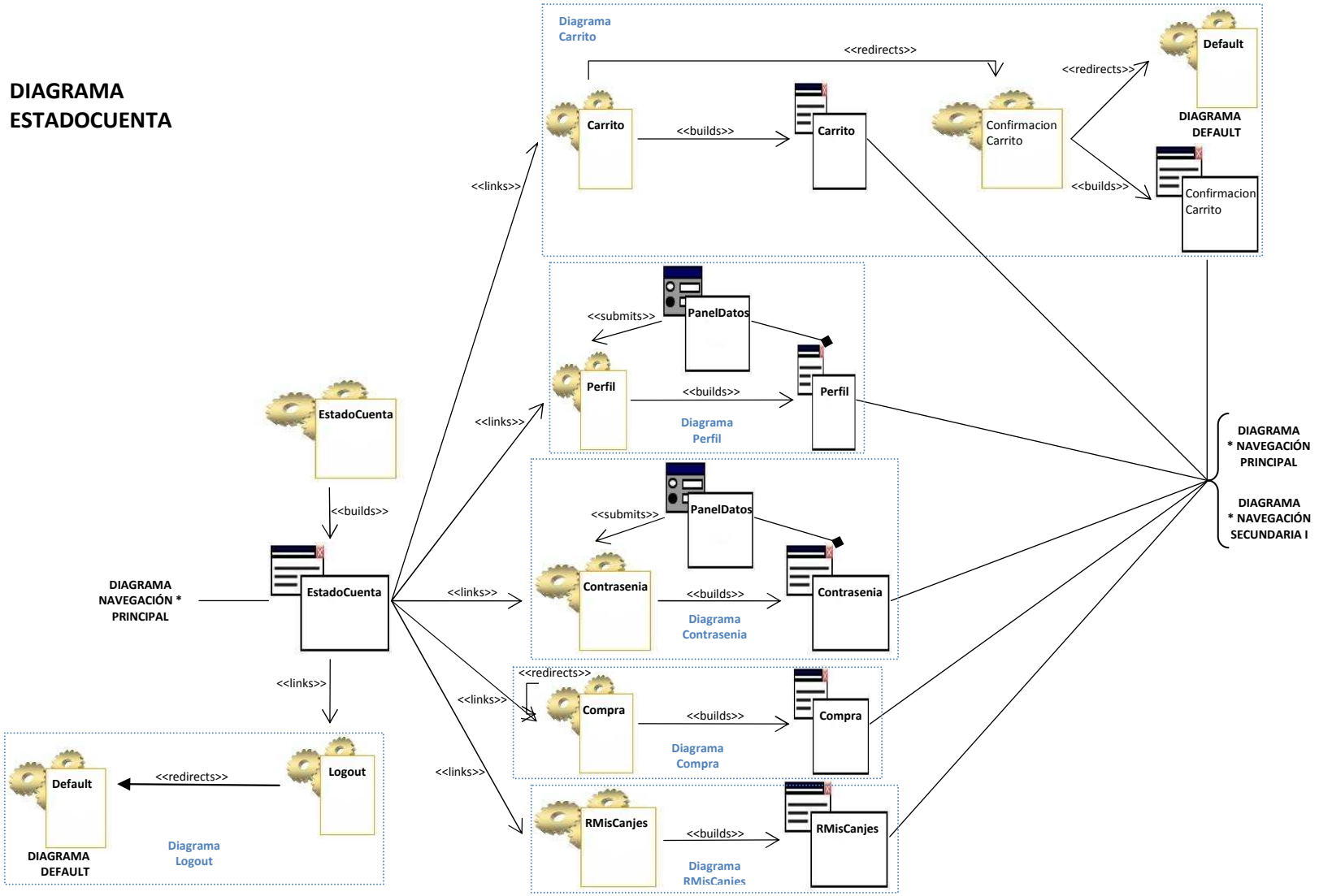

Figura 11.MDSU. Diagrama WAE “Estado de Cuenta”

A continuación los diagramas de casos de uso que se refinan en esta fase (nivel 2 y 3) con sus respectivas descripciones formales.

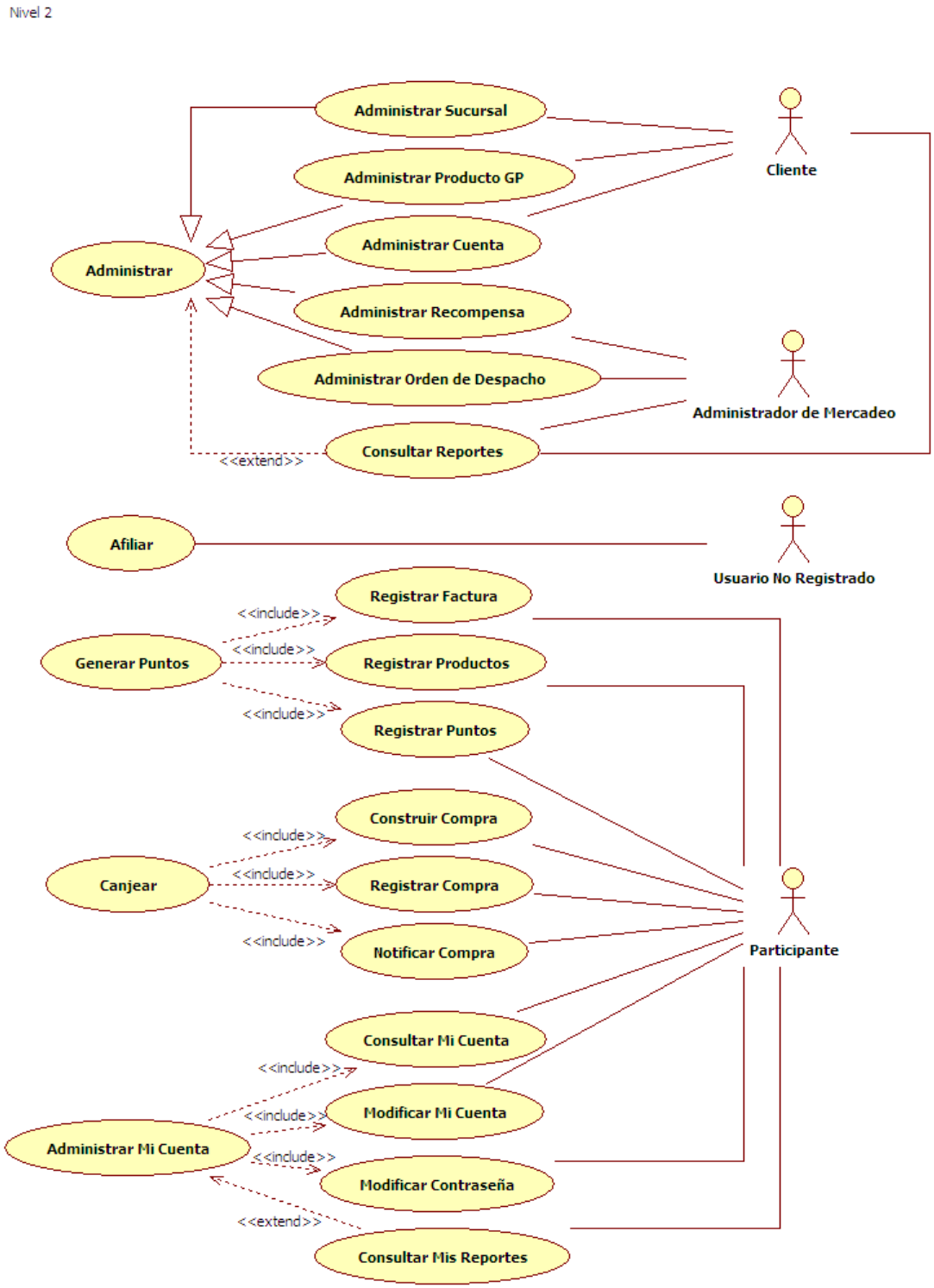

Figura 12.MDSU. Diagrama de casos de uso nivel 2

Descripción de los casos de uso

Nivel 2

Nombre: **Administrar Sucursal**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar el listado de sucursales del club.
Flujo Principal:

1. El sistema le permite al usuario administrar las sucursales pertenecientes al club, es decir crear, modificar o consultar una sucursal.

Flujo Alterno:
Post-Condición: Sucursal(es) administrada(s).

Nombre: **Administrar Producto GP** (Generador de Puntos).
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar el listado de productos generadores de puntos.
Flujo Principal:

1. El sistema le permite al usuario administrar los productos generadores de puntos del club, es decir crear, modificar o consultar un producto generador de puntos.

Flujo Alterno:
Post-Condición: Producto(s) GP administrado(s).

Nombre: **Administrar Cuenta**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar el listado de participantes.
Flujo Principal:

1. El sistema le permite al usuario administrar las cuentas de los participantes del club, es decir modificar o consultar una cuenta.

Flujo Alterno:
Post-Condición: Cuenta(s) administrada(s).

Nombre: **Administrar Recompensa**.
Actor: Administrador de mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar el listado recompensas.
Flujo Principal:

1. El sistema le permite al usuario administrar las recompensas ofrecidas en los clubes, es decir crear, modificar o consultar una recompensa.

Flujo Alterno:
Post-Condición: Recompensa(s) administrada(s).

Nombre: **Administrar Orden de Despacho**.
Actor: Administrador de mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar el listado órdenes de despacho.
Flujo Principal:

1. El sistema le permite al usuario administrar las órdenes de despacho registradas, es decir crear, modificar el estatus de las mismas o consultarlas.

Flujo Alterno:
Post-Condición: Orden(es) de despacho administrada(s).

Nombre: **Consultar Reportes**.
Actor: Administrador del club (cliente) y administrador de mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para desplegar algún reporte.
Flujo Principal:

1. El sistema le permite al usuario consultar reportes de estados de cuenta, recompensas canjeadas, ranking de participantes y ranking de sucursales.

Flujo Alterno:
Post-Condición: Reporte consultado.

Nombre: **Registrar Factura**.
Actor: Participante.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para cargar una factura.
Flujo Principal:

1. El participante llena el formulario para registrar la factura: fecha, número de la factura impreso y número de control.
2. El sistema verifica la completitud y correctitud de los datos ingresados:
    a. Correcto:
    2.1. Continúa en el caso de uso “Registrar Productos”.
    b. No correcto:
    2.1. El sistema le notifica al usuario el hecho mediante un mensaje de error en la misma página y vuelve al paso 1.

Flujo Alterno: El usuario decide no completar el formulario de registro por lo que la información no es registrada.
Post-Condición: Factura validada para registro.

Nombre: **Registrar Productos**.
Actor: Participante.
Pre-Condición: Los principales datos de la factura han sido validados.
Flujo Principal:

1. El participante indica los productos registrados en la factura así como sus respectivas cantidades pudiendo eliminar alguno en cualquier momento.
2. El participante pulsa el botón para registrar la factura.
3. El sistema registra la factura con sus respectivos productos, continúa en el caso de uso “Registrar Puntos”.

Flujo Alterno: El usuario decide cancelar la operación por lo que la información no es registrada.
Post-Condición: Factura completa registrada.

Nombre: **Registrar Puntos**.
Actor: Participante.
Pre-Condición: La factura completa ha sido registrada.
Flujo Principal:

1. El sistema calcula los puntos correspondientes por la factura.
2. El sistema registra la factura en la base de datos sumándole al participante los puntos correspondientes.
3. El sistema registra la generación de puntos en el archivo histórico.
4. El sistema notifica al usuario el éxito de la operación y finaliza.

Flujo Alterno:
Post-Condición: Puntos sumados al participante.
Nota: Tomar en cuenta que los puntos son repartidos dentro de los niveles de participantes según el porcentaje que le corresponde a cada uno.

Nombre: **Construir Compra**.
Actor: Participante.
Pre-Condición: El participante ha iniciado sesión y ha hecho click en el botón para canjear sus puntos.
Flujo Principal:

1. El participante selecciona la recompensa que desea y la agrega al carrito de compra. Repite este paso tantas veces como desee.
2. El participante pulsa en cualquiera de los enlaces para acceder a su carrito de compra actual.
3. El sistema le permite al usuario cambiar las cantidades de cada recompensa así como eliminarlas del carrito de compra según desee.
4. El participante pulsa el botón para actualizar su carrito de compra.
5. El sistema verifica que realmente pueda canjear la cantidad de puntos de que dispone por el(los) producto(s) seleccionado(s) (puntos>=∑(valor_producto*cantidad)):
   a. Válido:
   5.1. El participante vuelve al paso 3 o pulsa el botón para continuar con el canje, en caso tal continúa en el caso de uso “Registrar Compra”.
   b. No válido:
   5.2. El sistema notifica el hecho al participante mediante un mensaje de error en la misma página y vuelve al paso 3.

Flujo Alterno:
Post-Condición: Canje validado.

Nombre: **Registrar Compra**.
Actor: Participante.
Pre-Condición: Los principales datos del canje han sido validados.
Flujo Principal:

1. El sistema solicita al participante confirmar la dirección de envío de las recompensas.
2. El participante pulsa el botón para:
   a. Confirmar:
   2.1. El sistema registra el canje con sus correspondientes recompensas.
   2.2. El sistema crea una orden de compra con su respectiva orden de despacho asociándole la dirección de envío confirmada por el participante y las registra.
   2.3. El sistema actualiza los puntos del participante.
   2.4. El sistema registra el hecho en el archivo histórico.

2.5. Continúa en el caso de uso “Notificar Compra”.

b. Cancelar:

2.1. El sistema envía un correo electrónico al administrador del club notificándole el hecho.

2.2. El sistema vuelve a la página de inicio de la sesión y finaliza.

Flujo Alterno:

Post-Condición: Compra registrada y puntos redimidos, o cancelación notificada.

Nombre: **Notificar Compra**.

Actor: Participante.

Pre-Condición: El canje ha sido registrado.

Flujo Principal:

1. El sistema envía un correo electrónico al participante y al administrador de mercadeo notificando la operación de canje realizada.
2. El sistema notifica al usuario el éxito de la operación de canje en la misma página y finaliza.

Flujo Alterno: El participante nunca recibió el(los) producto(s).

Post-Condición: Participante satisfecho y premiado.

Nombre: **Consultar Mi Cuenta**.

Actor: Participante.

Pre-Condición: El participante ha iniciado sesión y ha pulsado en el botón o enlace para consultar sus datos.

Flujo Principal:

1. El sistema despliega los datos de cuenta de usuario: nombre completo, identificador personal, identificador de empresa, sucursal a la que pertenece, dirección, correo electrónico, teléfono de oficina y teléfono celular del participante en sesión.

Flujo Alterno:

Post-Condición: Datos de una cuenta consultados.

Nombre: **Modificar Mi Cuenta**.

Actor: Participante.

Pre-Condición: El participante ha iniciado sesión y ha pulsado en el botón o enlace para consultar sus datos.

Flujo Principal:

1. El sistema le permite al usuario modificar sus datos de comunicación como teléfonos y correo electrónico.
2. El participante modifica los datos que desea y pulsa el botón para registrar los cambios.
3. El sistema verifica la completitud y correctitud de los datos ingresados:
    a. Válido:
    3.1. El sistema registra los cambios, notifica el éxito de la operación en la misma página y finaliza.
    b. No válido:
    3.1. El sistema notifica el error en la misma página y vuelve al paso 2.

Flujo Alterno: El usuario decide cancelar la operación por lo que la información no es actualizada.

Post-Condición: Datos de una cuenta modificados.

Nombre: **Modificar Contraseña**.
Actor: Participante.
Pre-Condición: El participante ha iniciado sesión o se encuentra en la página de inicio del club.
Flujo Principal:

1. El sistema le permite al usuario modificar su contraseña, ya sea dentro de su sesión, de forma directa o, fuera de sesión, recordando su respuesta secreta o haciendo que el sistema genere una nueva contraseña y se la notifique mediante un correo electrónico.

Flujo Alterno:
Post-Condición: Contraseña modificada.

Nombre: **Consultar Mis Reportes**.
Actor: Participante.
Pre-Condición: El participante ha iniciado sesión y ha pulsado en el botón o enlace para consultar sus reportes.
Flujo Principal:

1. El sistema le permite al usuario ver reportes referentes a su estado de cuenta y a sus canjes realizados para realizarles seguimiento en caso de desearlo.

Flujo Alterno:
Post-Condición: Reportes consultados.

Figura 13.MDSU. Diagrama de casos de uso nivel 3 (Administración)

Descripción de los casos de uso

Nivel 3 (Administrar)

Nombre: **Crear Sucursal**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para crear una nueva sucursal.
Flujo Principal:

1. El sistema despliega al usuario un formulario solicitando los datos de la nueva sucursal.
2. El usuario llena el formulario y pulsa el botón para confirmar la creación de la nueva sucursal.
3. El sistema verifica la completitud y correctitud de los datos ingresados:
    a. Válido:
    3.1. El sistema registra la nueva sucursal, notifica el éxito de la operación en la misma página y finaliza.
    b. No válido:
    3.1. El sistema notifica el error en la misma página y vuelve al paso 2.

Flujo Alterno: El usuario decide cancelar la operación por lo que la información no es registrada.
Post-Condición: Nueva sucursal creada.

Nombre: **Modificar Estatus de Sucursal**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de sucursales.
Flujo Principal:

1. El sistema despliega al usuario el listado de sucursales.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la sucursal a la cual quiere cambiarle el estatus.
3. El sistema verifica el estatus de la misma para cambiarlo: de activa a desactiva y viceversa.

Flujo Alterno:
Post-Condición: Sucursal activada o desactivada.

Nombre: **Modificar Datos de Sucursal**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de sucursales.
Flujo Principal:

1. El sistema despliega al usuario el listado de sucursales.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la sucursal a la cual quiere modificarle los datos.
3. El sistema despliega al usuario un formulario editable con los datos actuales de la sucursal.
4. El usuario modifica el(los) campos que desea.
5. El usuario pulsa el botón para:
    a. Confirmar
    5.1.El sistema verifica la completitud y correctitud de los datos ingresados.
        a. Válido:

5.1.1. El sistema registra los cambios, notifica el éxito de la operación en la misma página y finaliza.

b. No válido:

5.1.1. El sistema notifica el error en la misma página y vuelve al paso 3.

b. Cancelar

5.1. El sistema no registra ninguna información y vuelve a la página del listado.

Flujo Alterno:

Post-Condición: Sucursal modificada.

Nombre: **Consultar Sucursal**.

Actor: Administrador del club (cliente).

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de sucursales.

Flujo Principal:

1. El sistema despliega al usuario el listado de sucursales.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la sucursal que quiere consultar.
3. El sistema despliega un formulario no editable con los datos de la sucursal.

Flujo Alterno:

Post-Condición: Sucursal consultada.

Nombre: **Crear Producto GP** (Generador de Puntos).

Actor: Administrador del club (cliente).

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para crear un nuevo producto.

Flujo Principal:

1. El sistema despliega al usuario un formulario solicitando los datos del nuevo producto.
2. El usuario llena el formulario y pulsa el botón para confirmar la creación del nuevo producto.
3. El sistema verifica la completitud y correctitud de los datos ingresados:
    a. Válido:

    3.1. El sistema registra el nuevo producto, notifica el éxito de la operación en la misma página y finaliza.

    b. No válido:

    3.2. El sistema notifica el error en la misma página y vuelve al paso 2.

Flujo Alterno: El usuario decide cancelar la operación por lo que la información no es registrada.

Post-Condición: Nuevo producto creado.

Nombre: **Modificar Estatus del Producto GP** (Generador de Puntos).

Actor: Administrador del club (cliente).

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de productos.

Flujo Principal:

1. El sistema despliega al usuario el listado de productos.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra el producto al cual quiere cambiarle el estatus.
3. El sistema verifica el estatus del mismo para cambiarlo: de activo a desactivo y viceversa.

Post-Condición: Producto activado o desactivado.

Nombre: **Modificar Datos del Producto GP** (Generador de Puntos).
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de productos.
Flujo Principal:

1. El sistema despliega al usuario el listado de productos.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra el producto al cual quiere modificarle los datos.
3. El sistema despliega al usuario un formulario editable con los datos actuales del producto.
4. El usuario modifica el(los) campos que desea.
5. El usuario pulsa el botón para:
    a. Confirmar
    5.1.El sistema verifica la completitud y correctitud de los datos ingresados.
        a. Válido:
        5.1.1. El sistema registra los cambios, notifica el éxito de la operación en la misma página y finaliza.
        b. No válido:
        5.1.1. El sistema notifica el error en la misma página y vuelve al paso 3.
    b. Cancelar
    5.1. El sistema no registra ninguna información y vuelve a la página del listado.

Flujo Alterno:
Post-Condición: Producto modificado.

Nombre: **Consultar Producto GP** (Generador de Puntos).
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de productos.
Flujo Principal:

1. El sistema despliega al usuario el listado de productos.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra el producto que quiere consultar.
3. El sistema despliega un formulario no editable con los datos del producto.

Flujo Alterno:
Post-Condición: Producto consultado.

Nombre: **Crear Recompensa**.
Actor: Administrador de Mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para crear una nueva recompensa.
Flujo Principal:

1. El sistema despliega al usuario un formulario solicitando los datos de la nueva recompensa.
2. El usuario llena el formulario y pulsa el botón para confirmar la creación de la nueva recompensa.
3. El sistema verifica la completitud y correctitud de los datos ingresados:
    a. Válido:
    3.1. El sistema registra la nueva recompensa, notifica el éxito de la operación en la misma página y finaliza.
    b. No válido:

3.3. El sistema notifica el error en la misma página y vuelve al paso 2.

Flujo Alterno: El usuario decide cancelar la operación por lo que la información no es registrada.

Post-Condición: Nueva recompensa creada.

Nombre: **Modificar Estatus de la Recompensa**.

Actor: Administrador de Mercadeo.

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de recompensas.

Flujo Principal:

1. El sistema despliega al usuario el listado de recompensas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la recompensa a la cual quiere cambiarle el estatus.
3. El sistema verifica el estatus de la misma para cambiarlo: de activa a desactiva y viceversa.

Flujo Alterno:

Post-Condición: Recompensa activada o desactivada.

Nombre: **Modificar Datos de la Recompensa**.

Actor: Administrador de Mercadeo.

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de recompensas.

Flujo Principal:

1. El sistema despliega al usuario el listado de recompensas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la recompensa a la cual quiere modificarle los datos.
3. El sistema despliega al usuario un formulario editable con los datos actuales de la recompensa.
4. El usuario modifica el(los) campos que desea.
5. El usuario pulsa el botón para:
    a. Confirmar

    5.1. El sistema verifica la completitud y correctitud de los datos ingresados.

    a. Válido:

    5.1.1. El sistema registra los cambios, notifica el éxito de la operación en la misma página y finaliza.

    b. No válido:

    5.1.1. El sistema notifica el error en la misma página y vuelve al paso 3.

    b. Cancelar

    5.1. El sistema no registra ninguna información y vuelve a la página del listado.

Flujo Alterno:

Post-Condición: Recompensa modificada.

Nombre: **Consultar Recompensa**.

Actor: Administrador de Mercadeo.

Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de recompensas.

Flujo Principal:

1. El sistema despliega al usuario el listado de recompensas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la recompensa que quiere consultar.

3. El sistema despliega un formulario no editable con los datos de la recompensa.

Flujo Alterno:
Post-Condición: Recompensa consultada.

Nombre: **Modificar Estatus de Cuenta**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de cuentas.
Flujo Principal:

1. El sistema despliega al usuario el listado de cuentas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la cuenta a la cual quiere cambiarle el estatus.
3. El sistema verifica el estatus de la misma para cambiarlo: de activa a desactiva y viceversa.

Flujo Alterno:
Post-Condición: Cuenta activada o desactivada.

Nombre: **Modificar Datos de Cuenta**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de cuentas.
Flujo Principal:

1. El sistema despliega al usuario el listado de cuentas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la cuenta a la cual quiere modificarle los datos.
3. El sistema despliega al usuario un formulario editable con los datos actuales de la cuenta.
4. El usuario modifica el(los) campos que desea.
5. El usuario pulsa el botón para:
    a. Confirmar

    5.1.El sistema verifica la completitud y correctitud de los datos ingresados.

    a. Válido:

    5.1.1. El sistema registra los cambios, notifica el éxito de la operación en la misma página y finaliza.

    b. No válido:

    5.1.1. El sistema notifica el error en la misma página y vuelve al paso 3.

    b. Cancelar

    5.1. El sistema no registra ninguna información y vuelve a la página del listado.

Flujo Alterno:
Post-Condición: Cuenta modificada.

Nombre: **Consultar Sucursal**.
Actor: Administrador del club (cliente).
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de cuentas.
Flujo Principal:

1. El sistema despliega al usuario el listado de cuentas.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la cuenta que quiere consultar.
3. El sistema despliega un formulario no editable con los datos de la cuenta.

Flujo Alterno:
Post-Condición: Cuenta consultada.

Nombre: **Modificar Estatus de la Orden de Despacho**.
Actor: Administrador de Mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de órdenes de despacho.
Flujo Principal:

1. El sistema despliega al usuario el listado de órdenes de despacho.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la orden de despacho a la cual quiere cambiarle el estatus.
3. El sistema verifica el estatus de la misma para cambiarlo al siguiente estatus (por enviar).

Flujo Alterno:
Post-Condición: Orden de despacho activada o desactivada.

Nombre: **Consultar Orden de Despacho**.
Actor: Administrador de Mercadeo.
Pre-Condición: El usuario ha iniciado sesión y ha hecho click en el botón o enlace para ver el listado de órdenes de despacho.
Flujo Principal:

1. El sistema despliega al usuario el listado de órdenes de despacho.
2. El usuario pulsa el botón o enlace de la fila en la que se encuentra la orden de despacho que quiere consultar.
3. El sistema despliega un formulario no editable con los datos de la orden de despacho.

Flujo Alterno:
Post-Condición: Orden de despacho consultada.

En cuanto al modelado de datos, a partir del diagrama entidad-relación generado en la primera fase, se crea la base de datos para almacenar la información necesaria en la aplicación. A continuación el diagrama físico de la base de datos por partes según la entidad principal.

Diagramas físicos de la base de datos

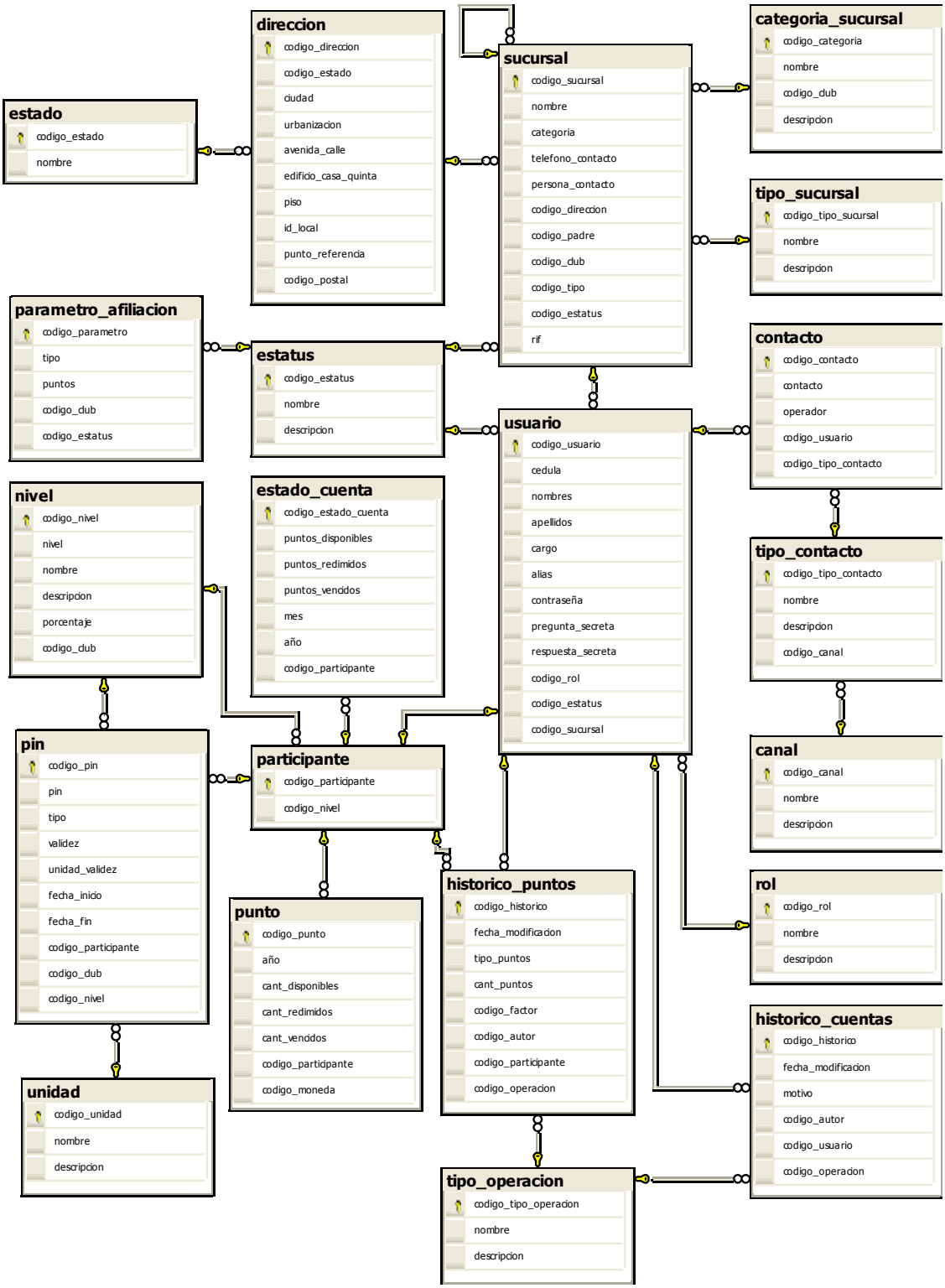

Figura 14.MDSU. Diagrama físico de la base de datos (Usuario)

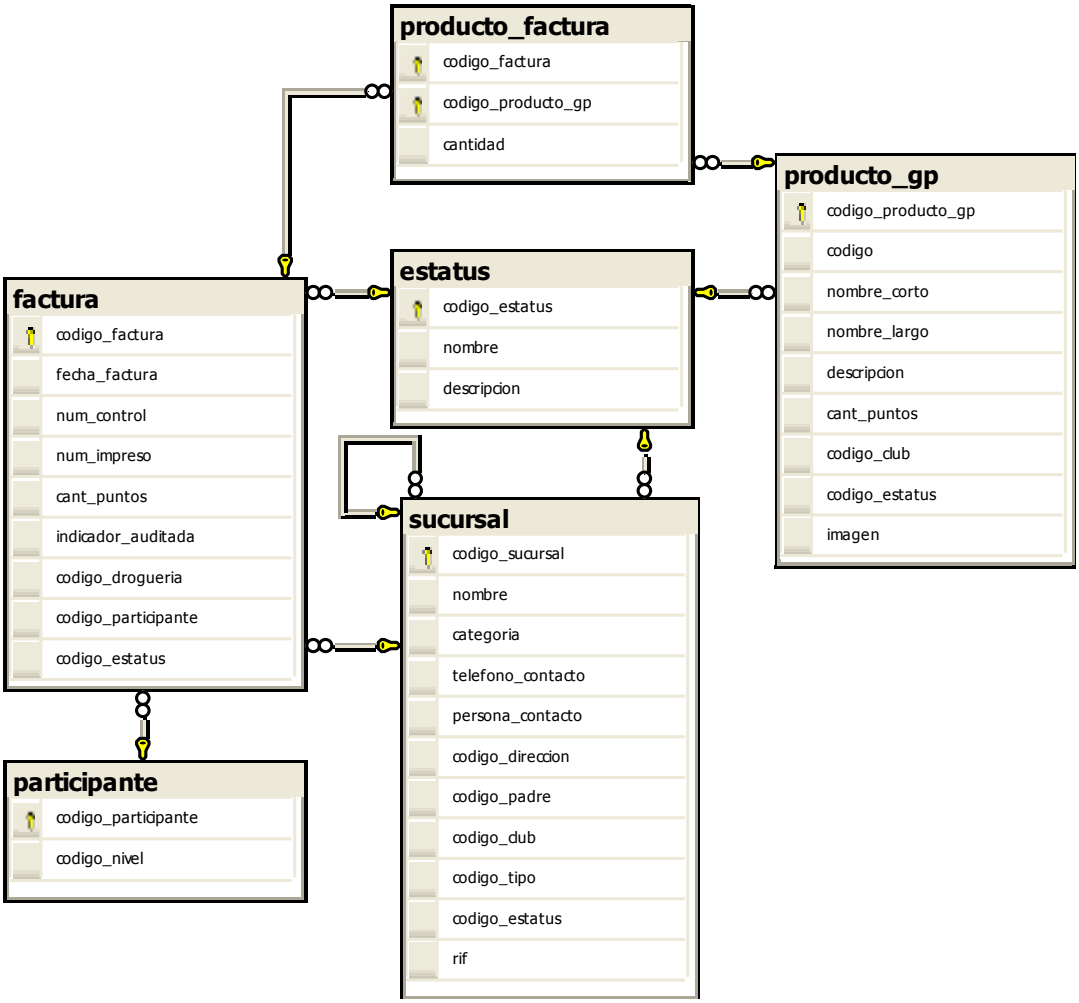

Figura 15.MDSU. Diagrama físico de la base de datos (Factura)

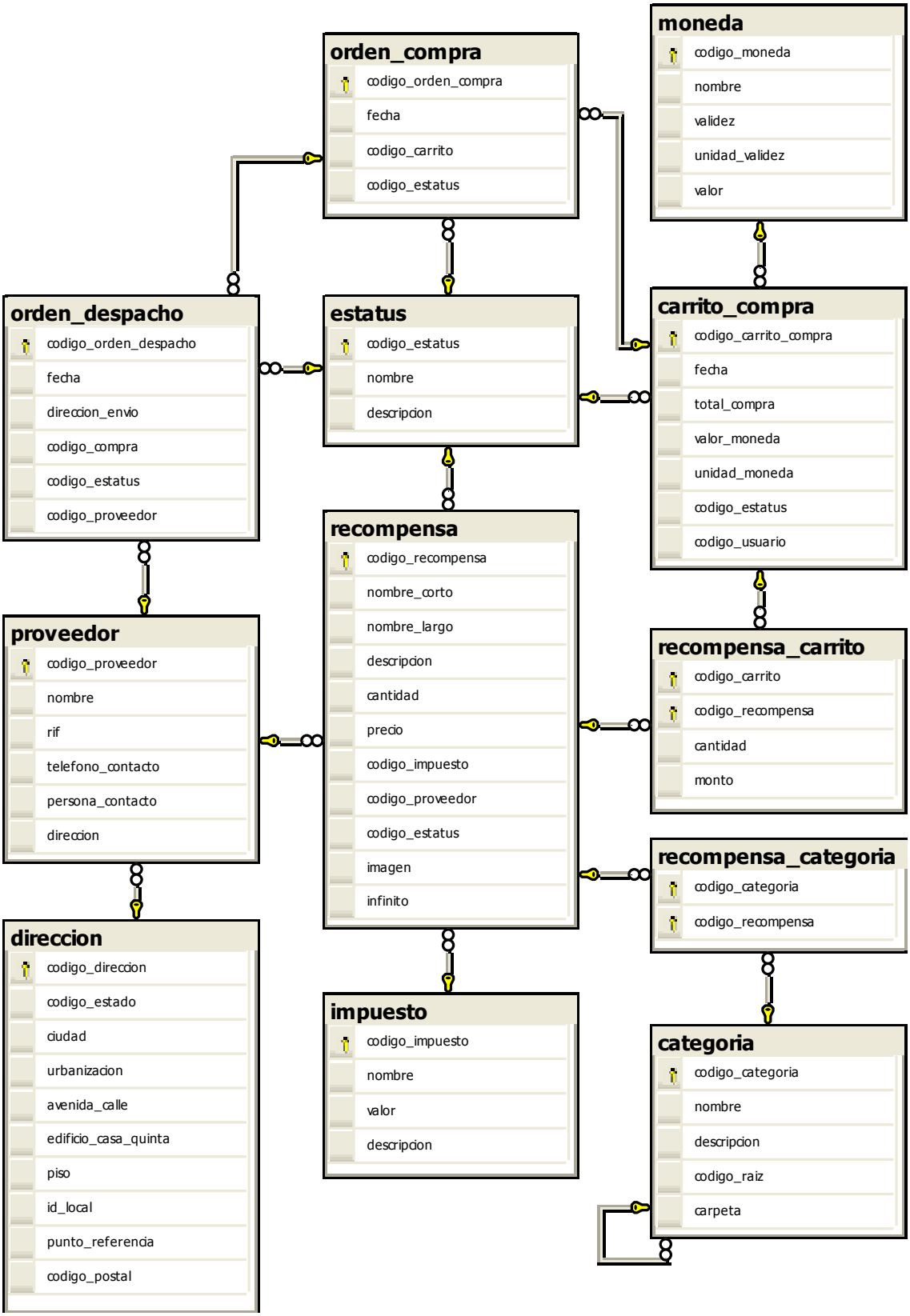

Figura 16.MDSU. Diagrama físico de la base de datos (Carrito de compra)

3.8.3. Fase de Construcción

En la fase de construcción se sigue la metodología Scrum dividiendo el trabajo en iteraciones de una semana en las cuales se seleccionaban funcionalidades que pudieran ser implementadas durante la iteración para así entregar al final de la misma un incremento funcional de la aplicación. Se concentra en la elaboración de un producto totalmente operativo y eficiente.

En esta fase se realizan siete (7) iteraciones por lo que la fase se extiende durante siete (7) (semanas).

A continuación se muestran de forma general las principales funcionalidades que se cubrieron en cada iteración a través de los diagramas de colaboración, secuencia y/o actividades de las mismas.

a. Iteración Nro. 1

En esta iteración se realiza todo lo referente al registro de un nuevo usuario al programa de incentivos y a la autenticación. A continuación los respectivos diagramas: diagrama de colaboración, diagrama de secuencia del sistema y diagrama de actividades de la funcionalidad "afiliación". El diagrama referente a la funcionalidad de autenticación se encuentran en el apartado "Anexos - Figura 17.A. Diagrama de actividades Autenticación".

Afiliación

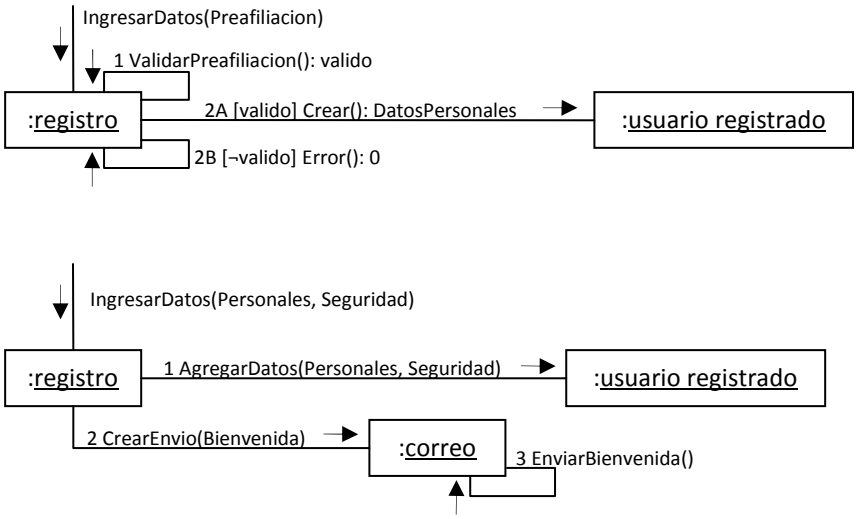

Figura 17.MDSU. Diagrama de colaboración "Afiliación"

Afiliar

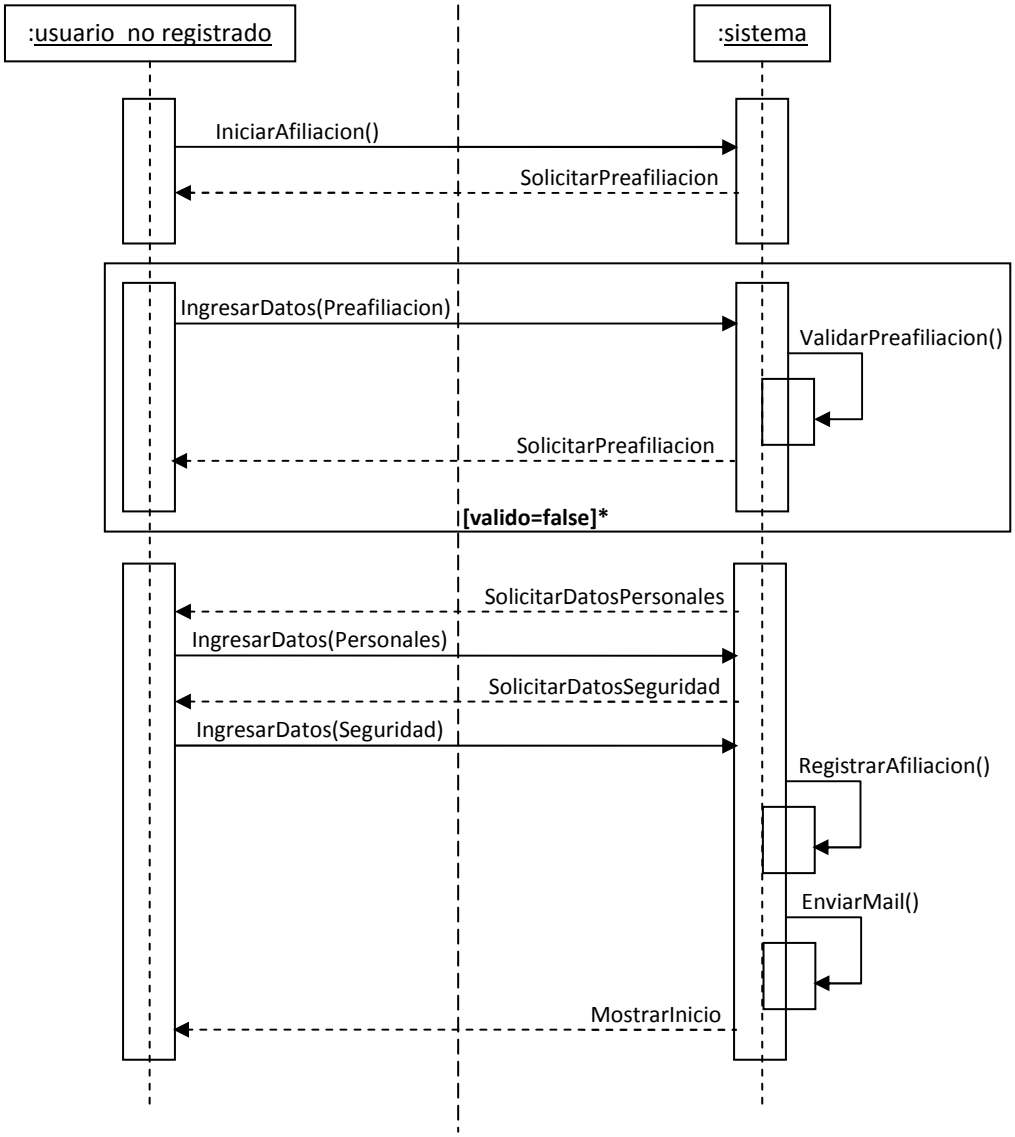

Figura 18.MDSU. Diagrama de secuencia “Afiliar”

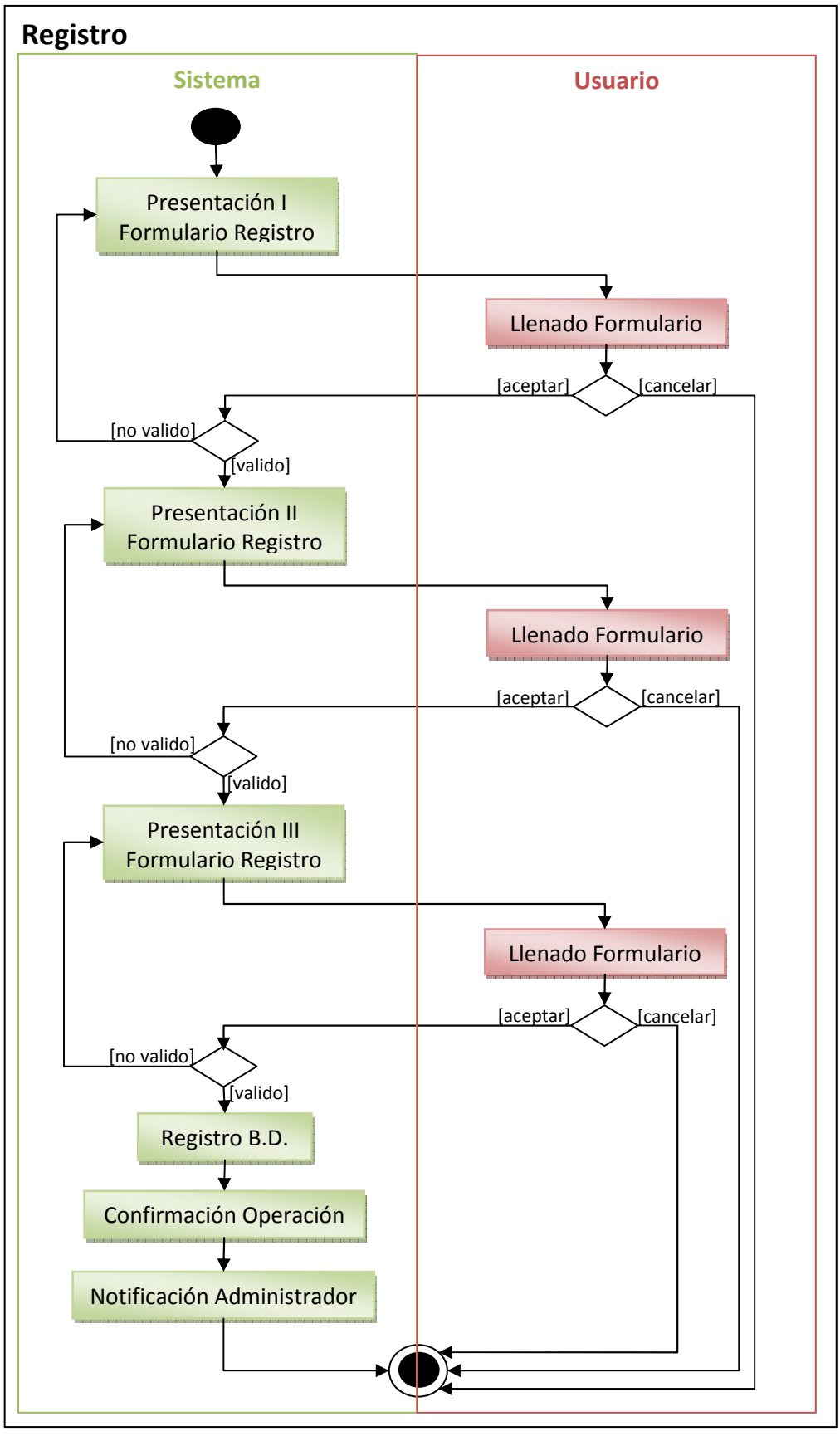

Figura 19.MDSU. Diagrama de actividades “Registro”

b. Iteración Nro. 2

Aquí se cubren las funcionalidades de registro de compras por parte de los participantes, incluyendo en la misma el registro de la factura, de los productos asociados a la factura y de los puntos, reflejados todas ellas en los siguientes diagramas: de colaboración y de secuencia del sistema. El diagrama de actividades correspondiente se encuentra en el apartado “Anexos - Figura 18.A. Diagrama de actividades Registro de Compra”.

Generación de Puntos

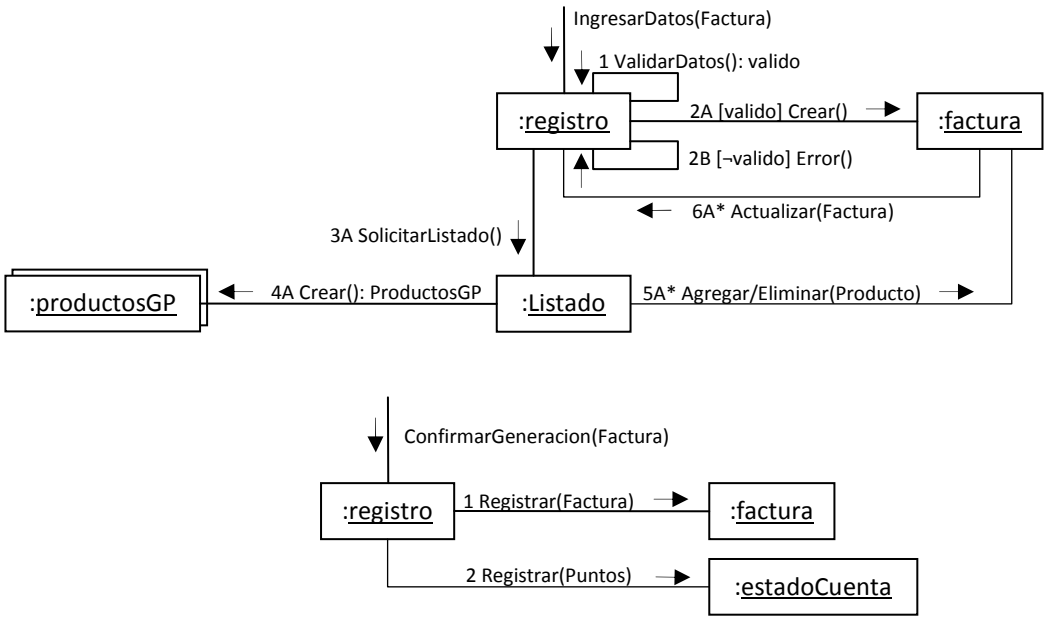

Figura 20.MDSU. Diagrama de colaboración “Generación de Puntos”

Generar Puntos

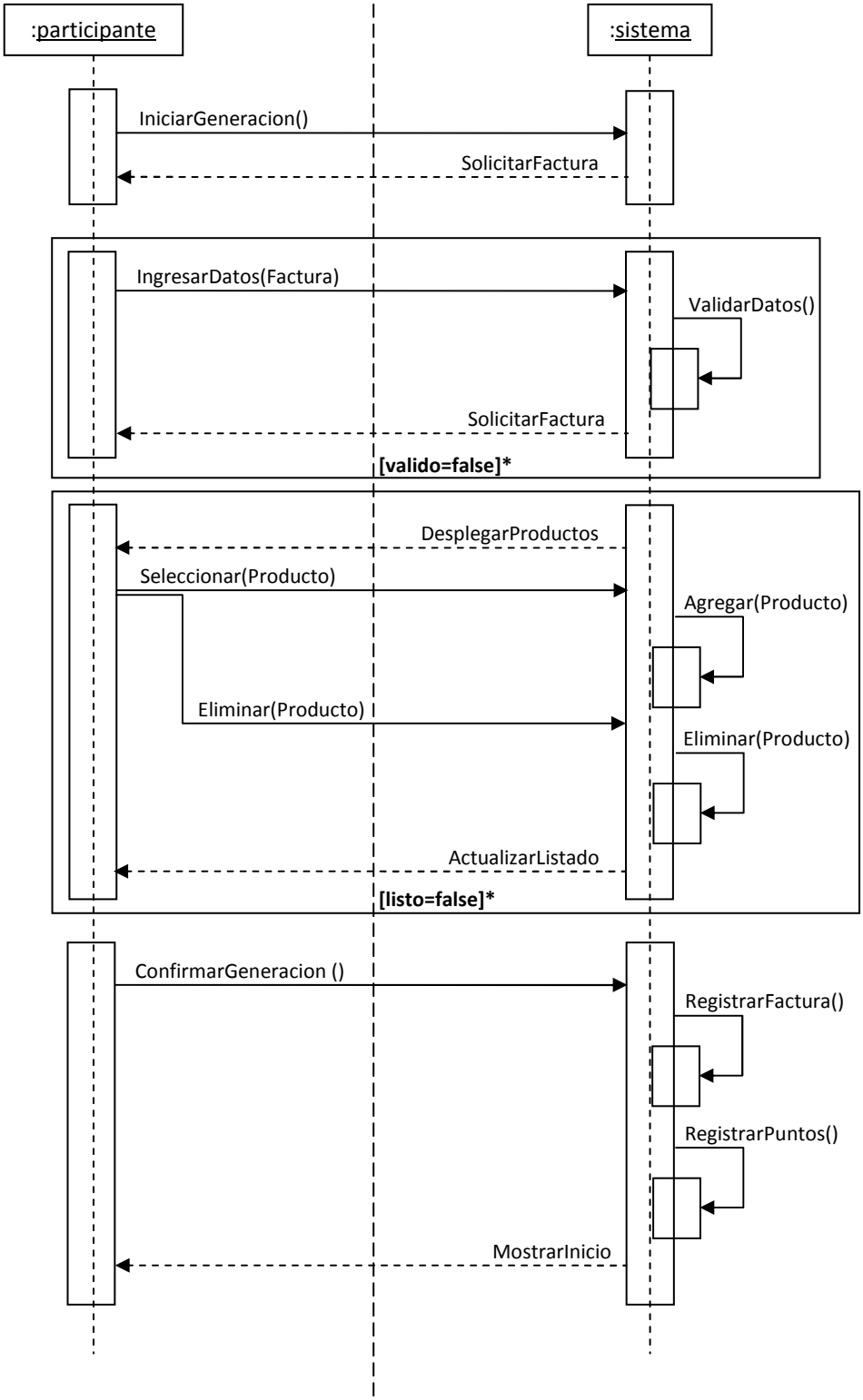

Figura 21.MDSU. Diagrama de secuencia “Generar Puntos”

c. Iteración Nro. 3

En esta iteración se implementa el carrito de compra en su totalidad. Esta funcionalidad se desglosa en: agregar o eliminar un producto al carrito de compra, actualizar la cantidad de un mismo producto dentro del carrito de compra y redimir efectivamente los puntos.

A continuación se presentan: el diagrama de colaboración y el diagrama de secuencia de la funcionalidad "redimir puntos" y "canjear", respectivamente, y los principales diagramas de actividades, el resto se encuentra en el apartado de "Anexos - Figura 19.A. Diagrama de actividades Eliminación de Producto(s) del Carrito de Compra y Figura 20.A. Diagrama de actividades Actualización de Cantidad de Producto(s) del Carrito de Compra".

Redención de Puntos

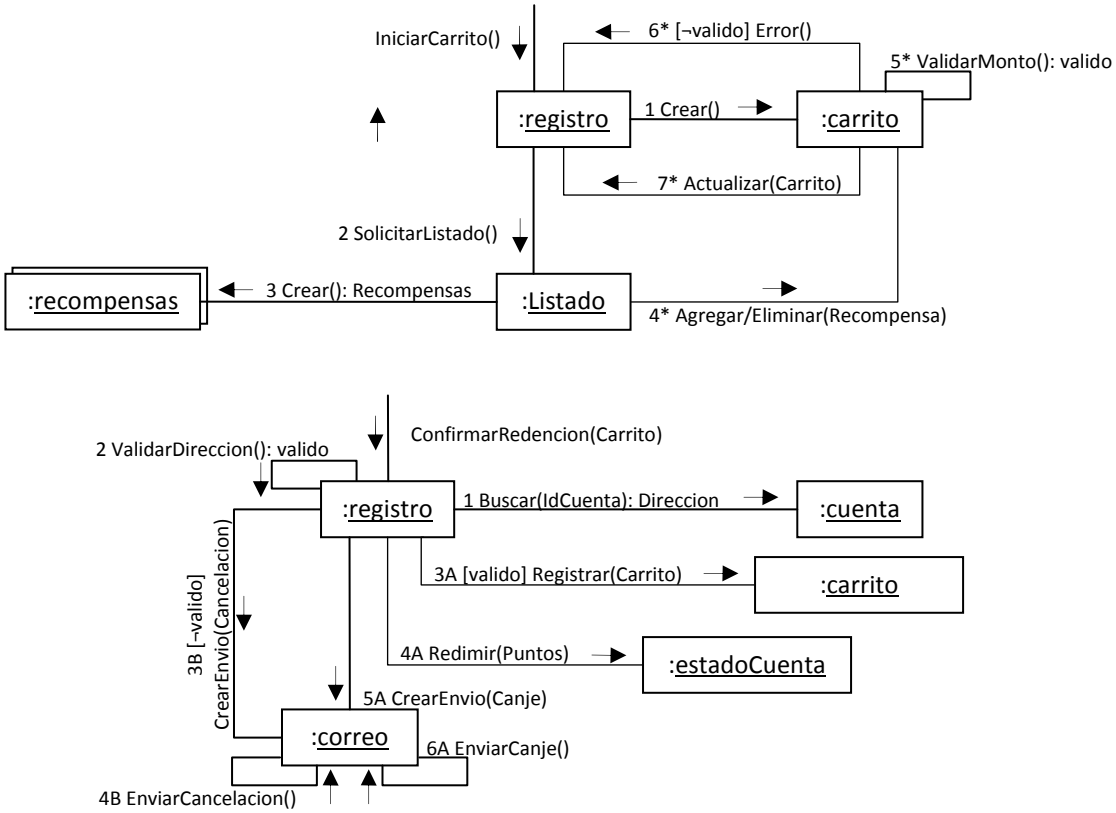

Figura 22.MDSU. Diagrama de colaboración "Redención de Puntos"

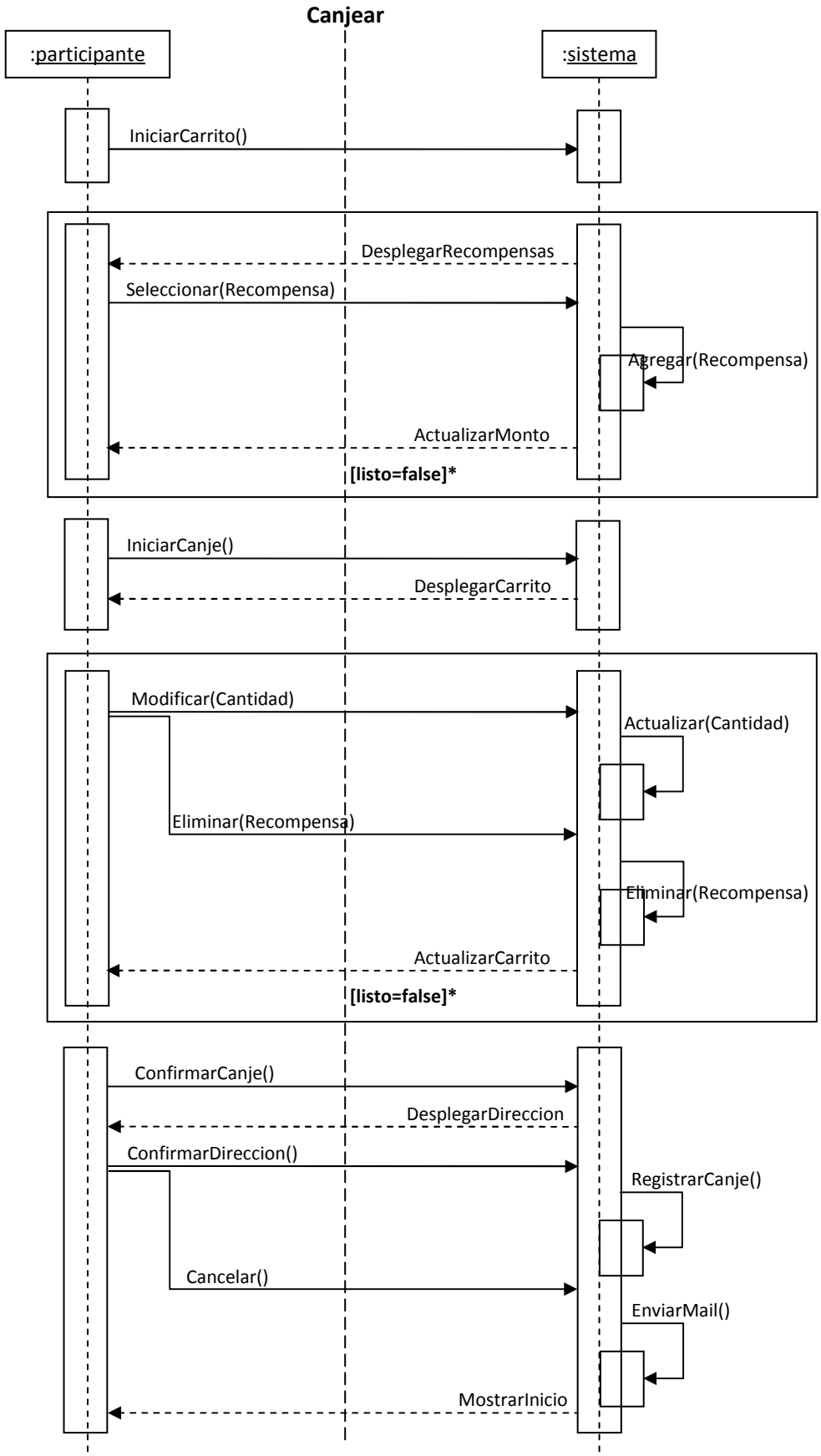

Figura 23.MDSU. Diagrama de secuencia “Canjear”

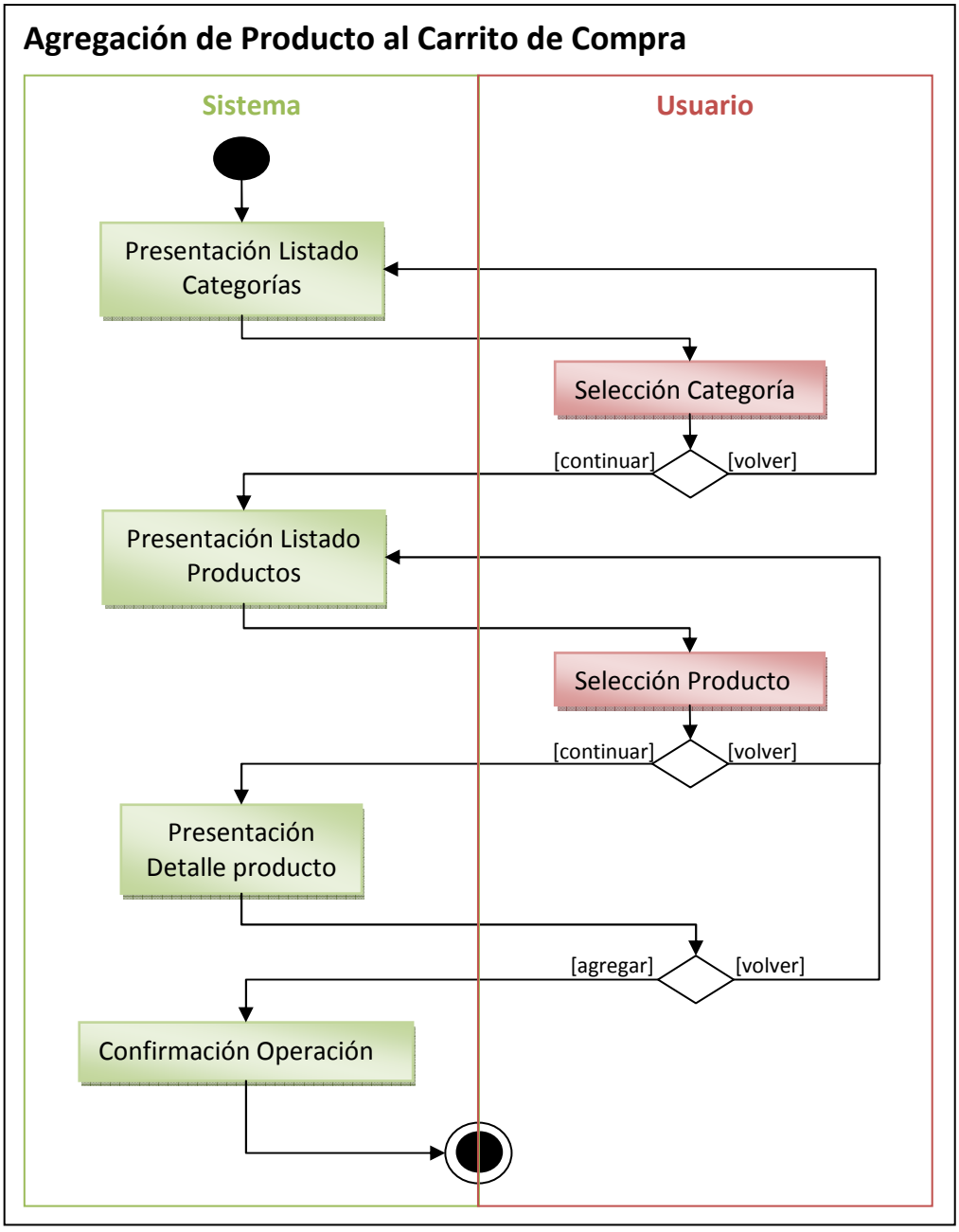

Figura 24.MDSU. Diagrama de actividades “Agregación de Producto al Carrito de Compra”

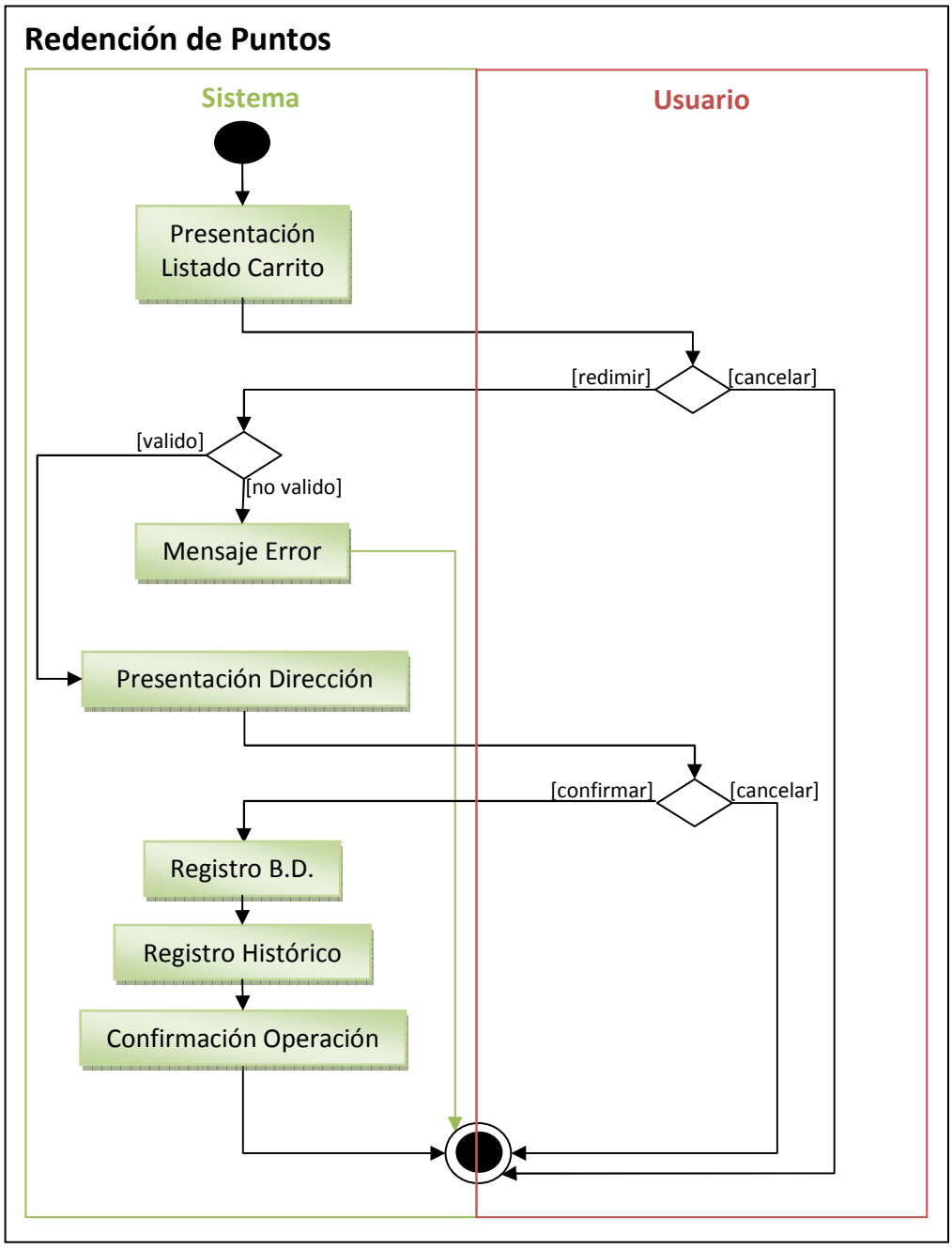

Figura 25.MDSU. Diagrama de actividades “Redención de Puntos”

d. Iteración Nro. 4

Las funcionalidades asociadas al cliente, visto como administrador del programa de incentivos, son implementadas en esta iteración. El cliente puede crear, consultar y modificar sucursales y productos participantes, además de consultar y modificar las cuentas de los participantes.

A continuación se muestran los diagramas de colaboración y los diagramas de secuencia del sistema para la administración de sucursales (creación, consulta y modificación), los demás diagramas, tanto de colaboración como de secuencia del sistema, referentes al resto de los elementos administrados por el cliente son muy análogos y se encuentran en el apartado de "Anexos - Figura 1.A. Diagrama de colaboración Creación de Producto GP, Figura 2.A. Diagrama de colaboración Consulta de Producto GP, Figura 3.A. Diagrama de colaboración Modificación de Producto GP, Figura 4.A. Diagrama de colaboración Consulta de Cuenta, Figura 5.A. Diagrama de colaboración Modificación de (Mi) Cuenta, Figura 9.A. Diagrama de secuencia Crear Producto GP, Figura 10.A. Diagrama de secuencia Consultar Producto GP, Figura 11.A. Diagrama de secuencia Modificar Producto GP, Figura 12.A. Diagrama de secuencia Consultar Cuenta y Figura 13.A. Diagrama de secuencia Modificar (Mi) Cuenta".

Diagramas de colaboración

Creación de Sucursal

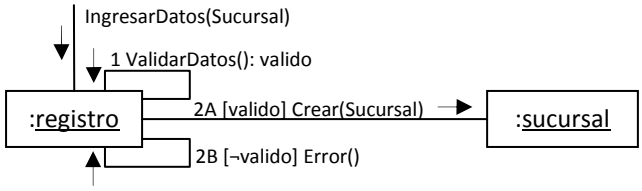

Figura 26.MDSU. Diagrama de colaboración "Creación de Sucursal"

Consulta de Sucursal

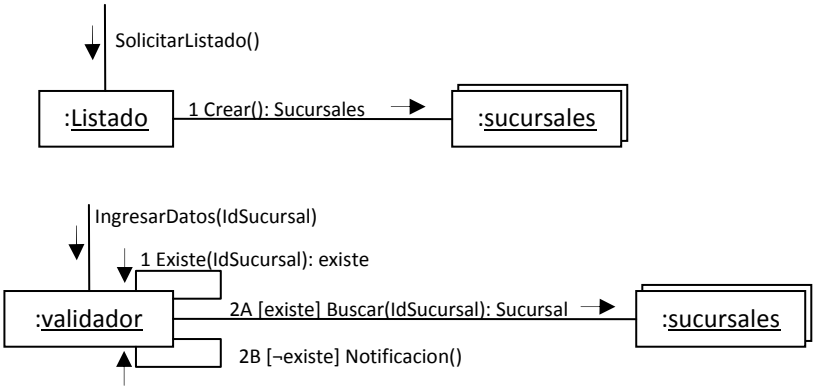

Figura 27.MDSU. Diagrama de colaboración "Consulta de Sucursal"

Modificación de Sucursal

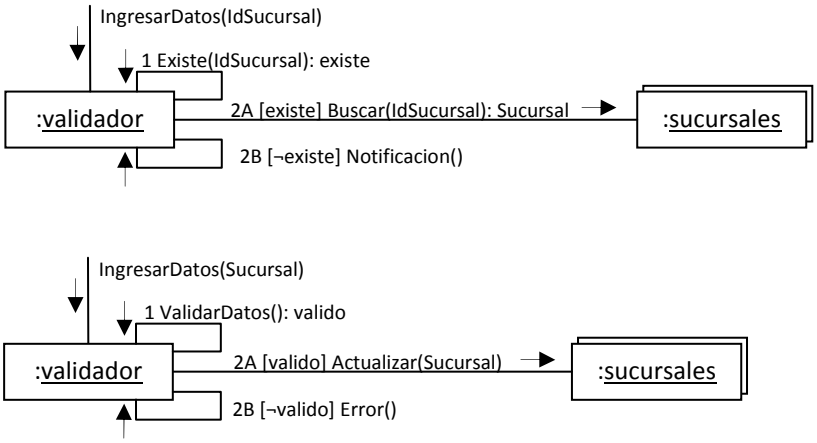

Figura 28.MDSU. Diagrama de colaboración "Modificación de Sucursal"

Diagramas de secuencia del sistema

Administrar ← Administrar Sucursal – Crear Sucursal

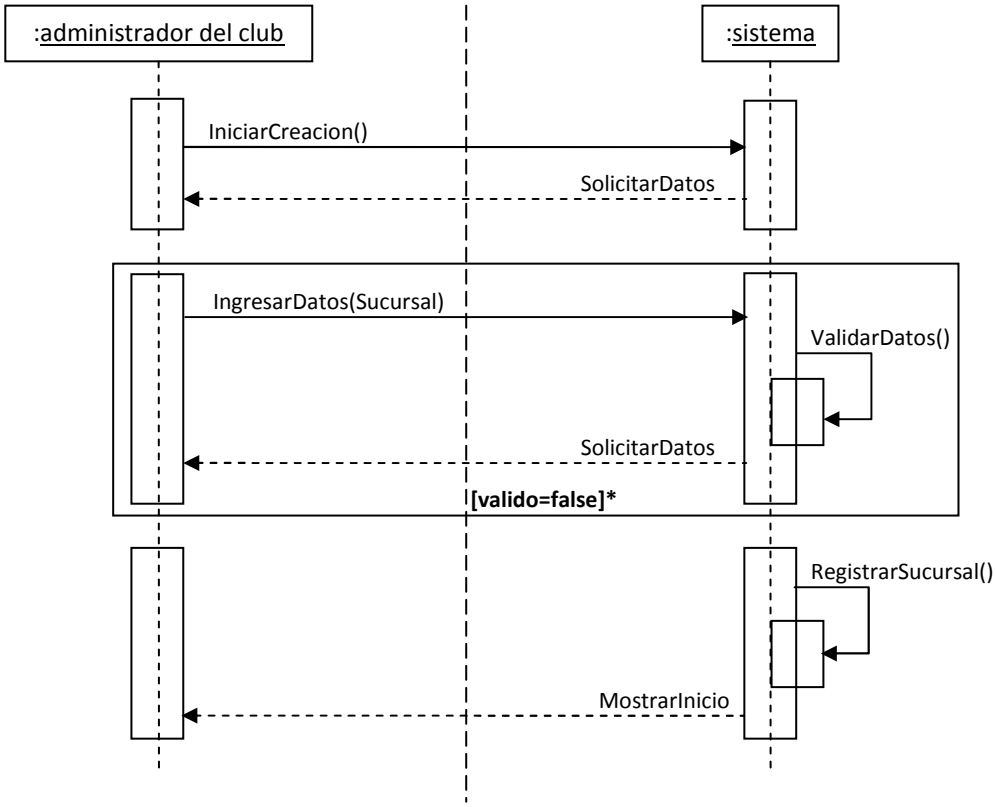

Figura 29.MDSU. Diagrama de secuencia "Crear Sucursal"

Administrar ← Administrar Sucursal – Consultar Sucursal

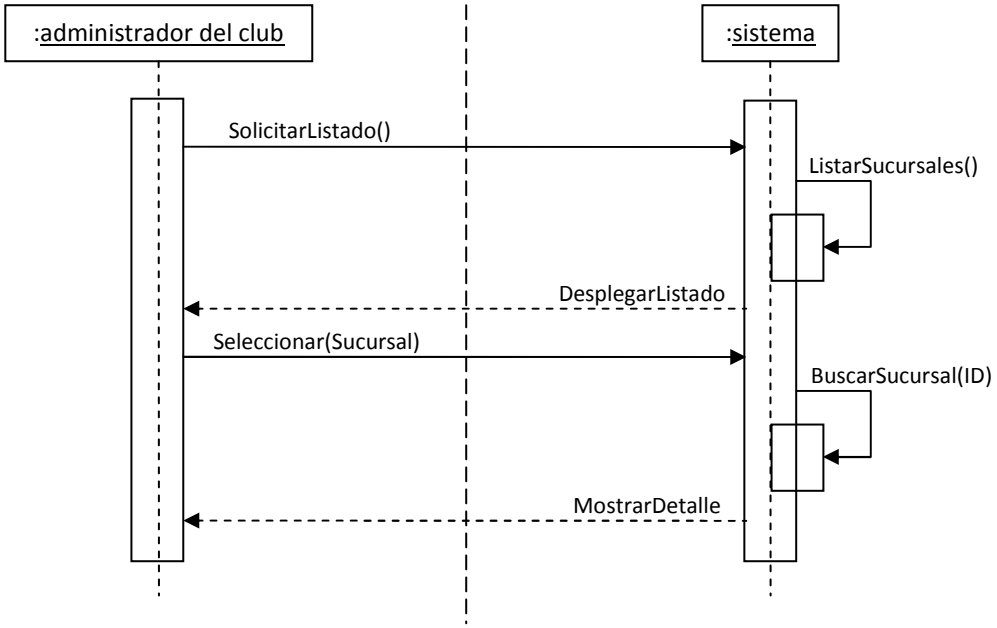

Figura 30.MDSU. Diagrama de secuencia "Consultar Sucursal"

Administrar ← Administrar Sucursal – Modificar Sucursal

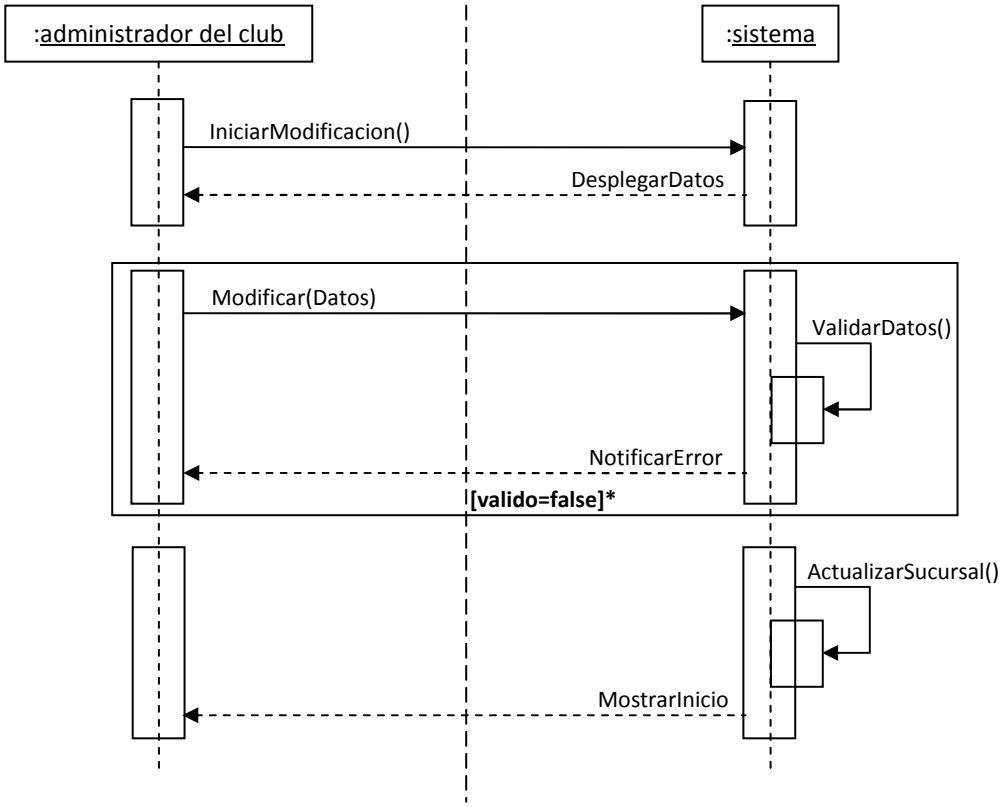

Figura 31.MDSU. Diagrama de secuencia "Modificar Sucursal"

e. Iteración Nro. 5

Siguiendo con las funcionalidades administrativas, en esta iteración se implementan las asociadas al administrador de mercadeo que incluyen crear, consultar y modificar recompensas y consultar y modificar el estatus de órdenes de despacho.

A continuación se muestran los diagramas de colaboración y los diagramas de secuencia del sistema para la administración de las recompensas; en cuanto al resto de los diagramas, referentes a la administración de las órdenes de despacho, se encuentran en el apartado "Anexos - Figura 6.A. Diagrama de colaboración Consulta de Orden de Despacho, Figura 7.A. Diagrama de colaboración Modificación del Estatus de Orden de Despacho, Figura 14.A. Diagrama de secuencia Consultar Orden de Despacho y Figura 15.A. Diagrama de secuencia Modificar Estatus de Orden de Despacho".

Diagramas de colaboración

Creación de Recompensa

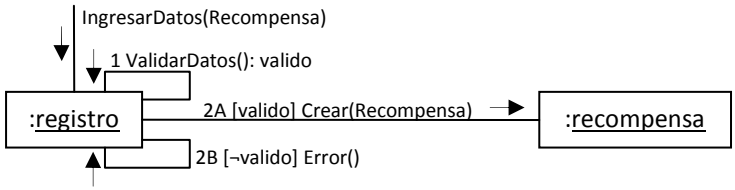

Figura 32.MDSU. Diagrama de colaboración "Creación de Recompensa"

Consulta de Recompensa

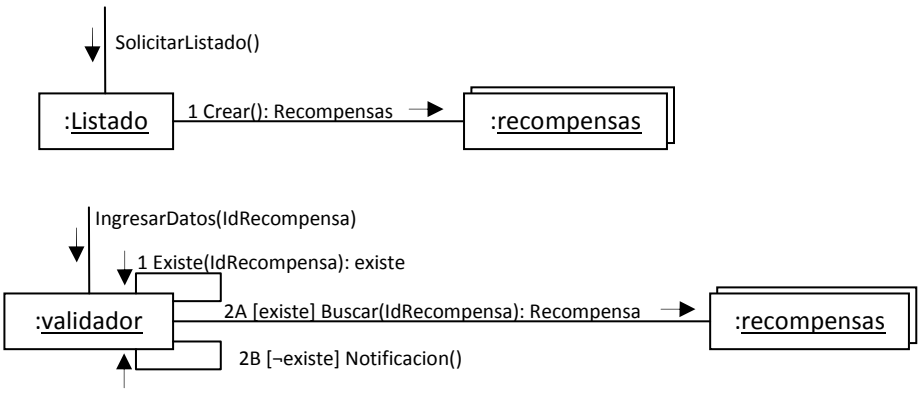

Figura 33.MDSU. Diagrama de colaboración "Consulta de Recompensa"

Modificación de Recompensa

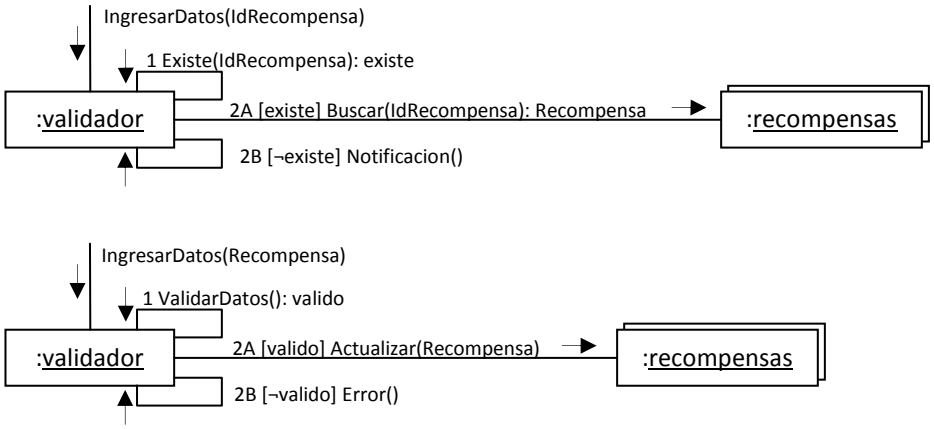

Figura 34.MDSU. Diagrama de colaboración "Modificación de Recompensa"

Diagramas de secuencia del sistema

Administrar ← Administrar Recompensa – Crear Recompensa

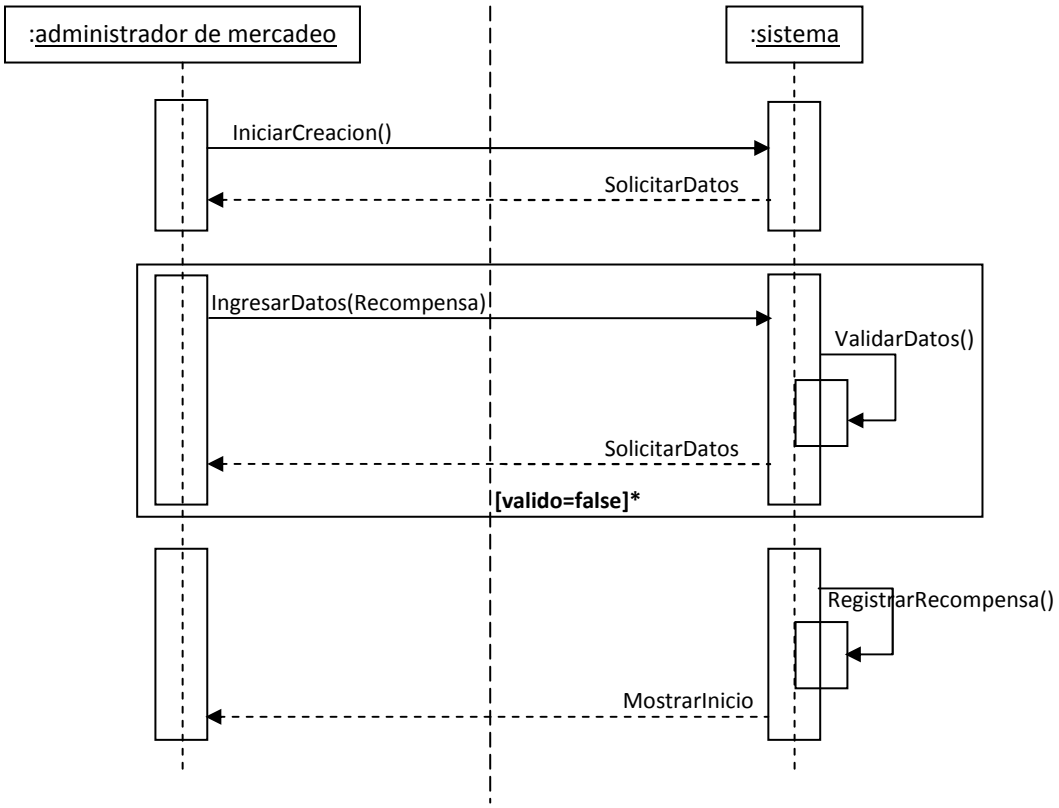

Figura 35.MDSU. Diagrama de secuencia "Crear Recompensa"

Administrar ← Administrar Recompensa – Consultar Recompensa

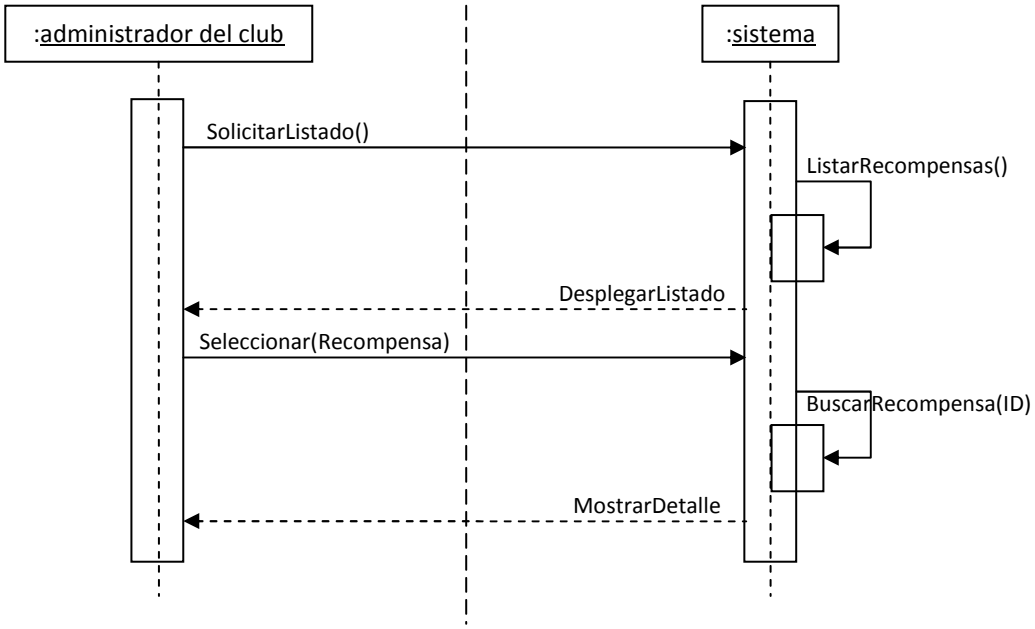

Figura 36.MDSU. Diagrama de secuencia “Consultar Recompensa”

Administrar ← Administrar Recompensa – Modificar Recompensa

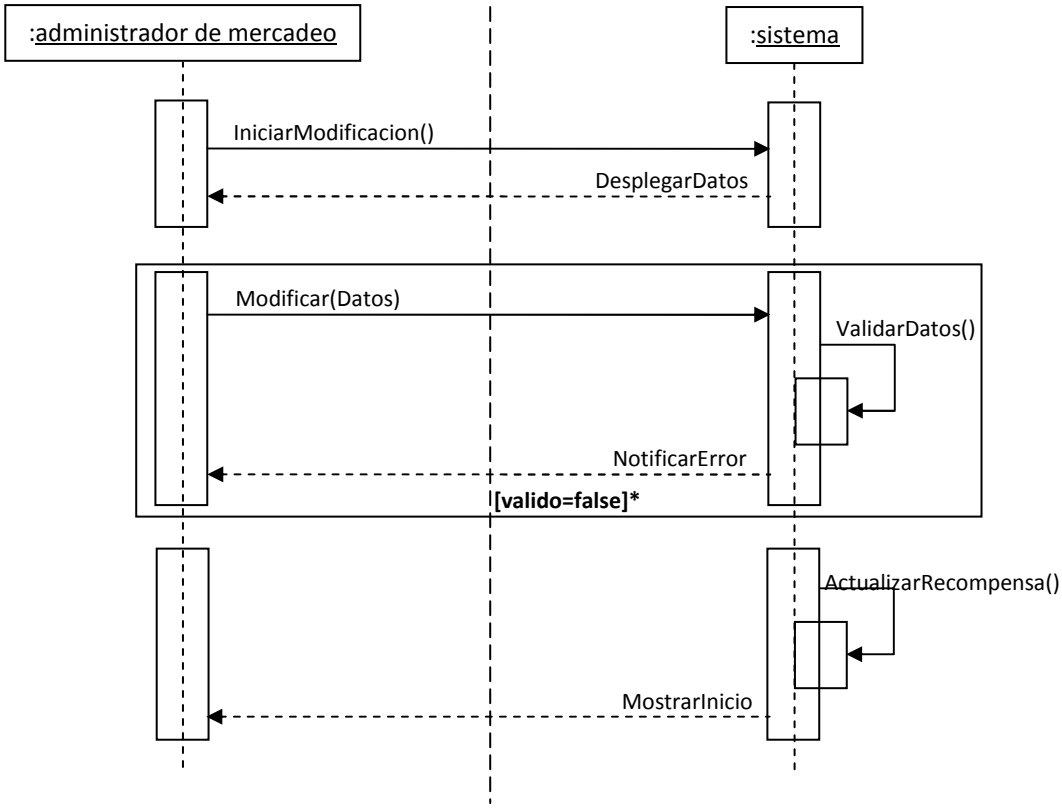

Figura 37.MDSU. Diagrama de secuencia “Modificar Recompensa”

f. Iteración Nro. 6

En esta iteración se implementan las funcionalidades referentes a la administración por parte de cada participante de su propia cuenta, es decir, la modificación de su cuenta y de su contraseña, por ello se muestran a continuación el diagrama de colaboración y el diagrama de secuencia del sistema referentes a la modificación de contraseña. Los diagramas de colaboración y secuencia asociados a la "consulta de mi cuenta" se encuentran en el apartado "Anexos - Figura 8.A. Diagrama de colaboración Consulta de Mi Cuenta y Figura 16.A. Diagrama de secuencia Consultar Mi Cuenta".

Modificación de Contraseña

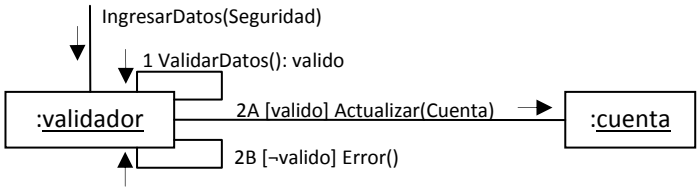

Figura 38.MDSU. Diagrama de colaboración "Modificación de Contraseña"

Administrar Mi cuenta – Modificar Contraseña

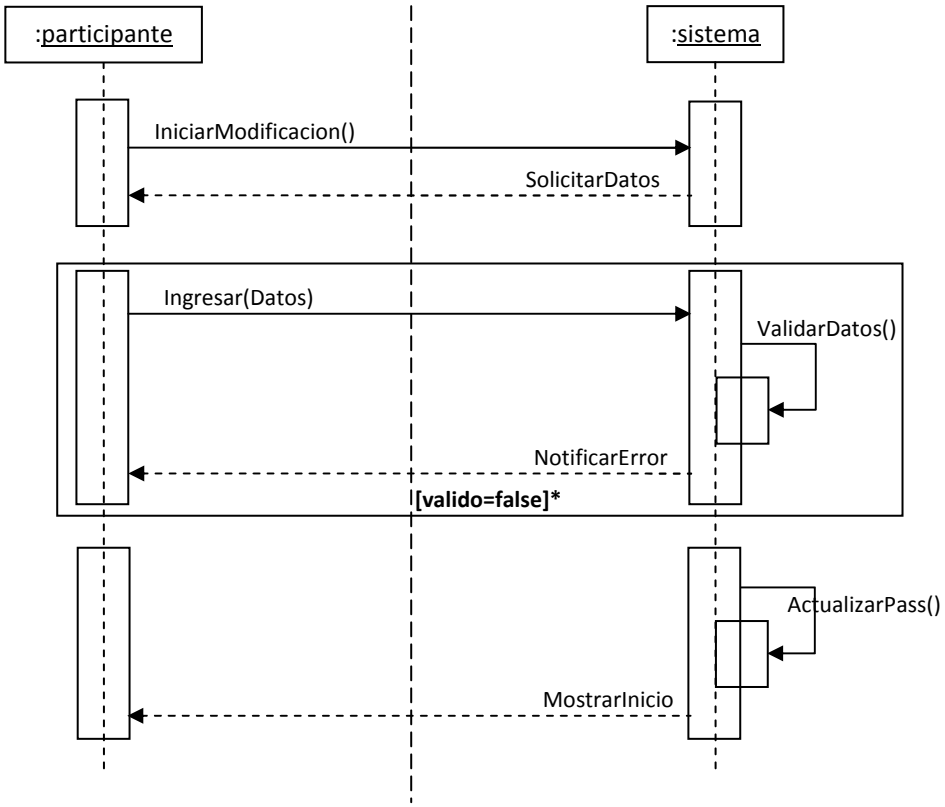

Figura 39.MDSU. Diagrama de secuencia "Modificar Contraseña"

g. Iteración Nro. 7

Para finalizar la implementación del módulo administrativo en esta última fase se desarrollan los reportes administrativos para cada perfil. El cliente necesita reportes referentes a las compras registradas, canjes realizados y rankings de participantes y sucursales. El administrador de mercadeo visualiza los canjes realizados y las órdenes de despacho generadas por los mismos. Y finalmente el participante visualiza su estado de cuenta y sus canjes realizados.

A continuación el diagrama de colaboración y el diagrama de secuencia del sistema para la consulta de reportes.

Consulta de (Mis) Reportes

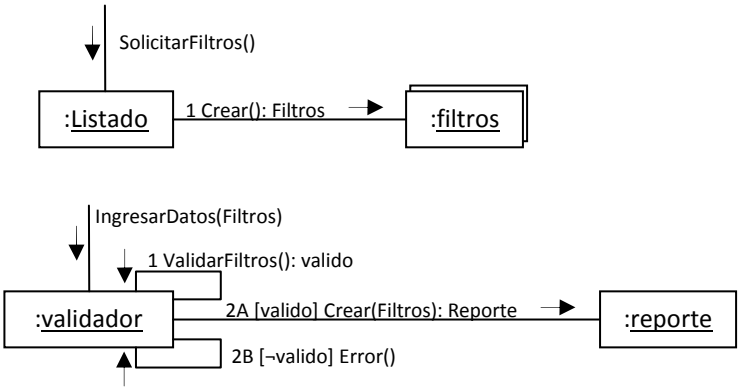

Figura 40.MDSU. Diagrama de colaboración "Consulta de (Mis) Reportes"

Administrar ← Administrar (Mi Cuenta) – Consultar (Mis) Reportes

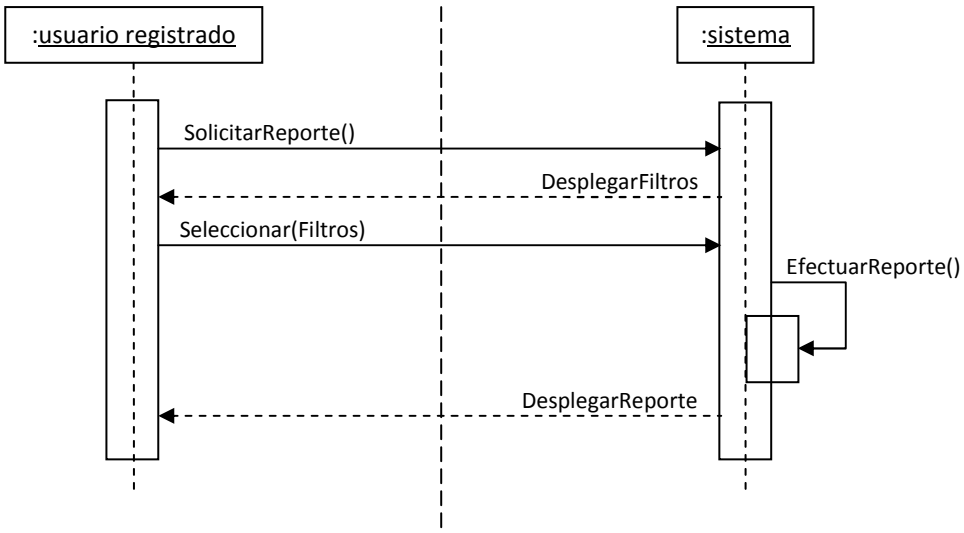

Figura 41.MDSU. Diagrama de secuencia "Consultar (Mis) Reportes"

3.8.4. Fase de Transición

En esta fase se instala la aplicación integrándose a la plataforma adecuada y se realizan las pruebas colectivas al sistema mediante una matriz de pruebas para garantizar el correcto funcionamiento del sistema, y se extrae un diccionario de datos para futuros mantenimientos de la base de datos. Esta fase se extiende por quince (15) días.

A continuación se muestra el diccionario de datos.

Canal: (tabla canal) Entidad débil que representa el canal por el cual es contactado un usuario (llamada telefónica, mensaje de texto, correo electrónico)
Atributos:

- codigo_canal: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre del canal (teléfono, sms, ...), varchar(20), no nulo
- descripcion: breve descripción explicativa del canal, varchar(50), nulo

Tipo Contacto: (tabla tipo_contacto) Entidad débil que representa el tipo de contacto de un usuario (teléfono principal, teléfono casa, teléfono trabajo, email principal, email secundario)
Atributos:

- codigo_tipo_contacto: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre del tipo de contacto (teléfono principal, teléfono secundario, ...), varchar(20), no nulo
- descripción: breve descripción explicativa del tipo de contacto, varchar(50), nulo
- codigo canal: clave foránea hacia la tabla canal, representa el canal que usa el tipo se contacto, entero, no nulo

Contacto: (tabla contacto) Entidad que representa los contactos (números telefónicos y correos electrónicos) que posee un usuario
Atributos:

- codigo_contacto: clave primaria de la tabla, entero, no nulo, aplica identidad
- contacto: identificador principal del contacto (5234678, luisa.jimenez, ...) , varchar(50), no nulo
- operador: operador del contacto (0412, @hotmail.com) , varchar(50), no nulo
- codigo_usuario: clave foránea hacia la tabla usuario, representa el usuario al cual pertenece el contacto, entero, no nulo
- codigo_tipo_contacto: clave foránea hacia la tabla tipo_contacto, representa el tipo de contacto, entero, no nulo

Rol: (tabla rol) Entidad que representa el rol que el usuario posee dentro del sistema (administrador general, administrador de club, participante)
Atributos:

- codigo_rol: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre del rol (administrador, participante, ...), varchar(30), nulo
- descripcion: breve descripción explicativa del rol, varchar(50), nulo

Estatus: (tabla estatus) Entidad que representa el estatus actual que posee un usuario, una sucursal, una factura, un carrito de compras, una orden de despacho, ... (activo, en proceso, aprobado, rechazado, bloqueado, finalizado, ...)
Atributos:

- codigo_estatus: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre del estatus (activo, rechazado, ...), varchar(20), no nulo
- descripcion: breve descripción explicativa del estatus y a qué entidad aplica (usuario, recompensa, ...), varchar(50), nulo

Usuario: (tabla usuario) Entidad fuerte que contiene los datos principales de los usuarios
Aributos:

- codigo_usuario: clave primaria de la tabla, entero, no nulo, aplica identidad
- alias: identificador con el cual el usuario ingresa al sistema como registrado, varchar(15), no nulo
- nombre_saludo: nombre que el sistema utiliza para referirse al usuario, varchar(75), no nulo
- contrasenia: contraseña que el usuario utiliza para entrar al sistema como registrado, y que el sistema utiliza para autenticarlo, varchar(8), no nulo
- pregunta_secreta: pregunta que el usuario establece para autenticación dentro del sistema, varchar(50), no nulo
- respuesta_secreta: respuesta que el usuario establece para autenticación dentro del sistema, varchar(50), no nulo
- codigo_rol: clave foránea hacia la tabla rol, representa el rol que tiene el usuario dentro del sistema, entero, no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual del usuario en el sistema, entero, no nulo

Participante: (tabla participante) Entidad fuerte que contiene los datos de los participantes
Atributos:

- codigo_participante: clave primaria de la tabla, entero, no nulo, aplica identidad
- ci_rif: cédula de identidad o rif del participante, varchar(10), no nulo
- codigo_carnet: identificador del participante dentro de la empresa, varchar(10), no nulo
- nombre_razon: nombre completo o razón social del participante, varchar(75), no nulo
- cargo: cargo que desempeña el participante dentro de la empresa, varchar(50), no nulo
- codigo_nivel: clave foránea hacia la tabla nivel, representa el nivel del participante, entero, no nulo
- codigo_sucursal: clave foránea hacia la tabla sucursal, representa la sucursal en la cual labora el participante, entero, no nulo
- codigo_usuario: clave foránea hacia la tabla usuario, representa el usuario con el cual el participante ingresa al sistema, entero, no nulo

Nivel: (tabla nivel) Entidad débil que representa los niveles de participantes, así como el porcentaje obtendido por cada uno en la generación de puntos
Atributos:

- codigo_nivel: clave primaria de la tabla, entero, no nulo, aplica identidad
- nivel: código del nivel establecido dentro del sistema, único, entero, no nulo
- nombre: nombre del nivel (propietario, vendedor, ...), varchar(30), no nulo
- descripcion: breve descripción explicativa del nivel, varchar(50), nulo

- porcentaje: porcentaje que el participante obtiene en la generación de puntos, decimal(4, 2), no nulo

Sucursal: (tabla sucursal) Entidad que representa las diferentes sucursales participantes en el club
Atributos:
- codigo_sucursal: clave primaria de la tabla, entero, no nulo, aplica identidad
- razon_social: razón social de la sucursal, varchar(50), no nulo
- rif: rif de la sucursal, varchar(20), nulo
- telefono_contacto: teléfono de contacto con la sucursal, varchar(20), nulo
- persona_contacto: persona de contacto con la sucursal, varchar(30), nulo
- codigo_direccion: clave foránea hacia la tabla direccion, representa la direccion de la sucursal, entero, no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual de la sucursal en el sistema, entero, no nulo

Estado Cuenta: (tabla estado_cuenta) Entidad que representa el estado de cuenta mensual de cada participante
Atributos:
- codigo_estado_cuenta: clave primaria de la tabla, entero, no nulo, aplica identidad
- disponibles: puntos disponibles del mes, decimal(10, 2), no nulo, valor por defecto 0.0
- redimidos: puntos canjeados del mes, decimal(10, 2), no nulo, valor por defecto 0.0
- vencidos: puntos vencidos del mes, decimal(10, 2), no nulo, valor por defecto 0.0
- mes: mes del estado de cuenta, entero, no nulo
- anio: año del estado de cuenta, entero, no nulo
- codigo_participante: clave foránea hacia la tabla participante, representa el participante al cual pertenece el estado de cuenta, entero, no nulo

Factura: (tabla factura) Entidad fuerte que contiene los principales datos de las facturas registradas en el sistema por parte de los participantes
Atributos:
- codigo_factura: clave primaria de la tabla, entero, no nulo, aplica identidad
- fecha_factura: fecha de emisión de la factura, datetime, no nulo
- numero_impreso: código de la factura, entero, no nulo
- numero_control: código de control de la factura, entero, no nulo
- cantidad_puntos: cantidad de puntos que genera la factura, decimal(10, 2), no nulo
- valor_bs_moneda: valor de la moneda al momento del registro de la factura, decimal(4, 2), no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual de la factura en el sistema, entero, no nulo
- codigo_participante: clave foránea hacia la tabla participante, representa el participante al cual pertenece la factura y por lo tanto los puntos generados, entero, no nulo

Producto GP (Generador de Puntos): (tabla producto_gp) Entidad fuerte que representa cada uno de los productos válidos en el club por los cuales los participantes pueden obtener puntos
Atributos:
- codigo_producto_gp: clave primaria de la tabla, entero, no nulo, aplica identidad
- codigo: código del producto dentro de la empresa, varchar(10), no nulo
- nombre_corto: nombre corto del producto, varchar(20), no nulo

- nombre_largo: nombre completo del producto, varchar(50), no nulo
- descripcion: descripción del producto, varchar(500), no nulo
- cantidad_puntos: cantidad de puntos que genera el producto por unidad, entero, no nulo
- imagen: ruta de ubicación de la imagen del producto, varchar(150), no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual del producto en el sistema, entero, no nulo

Relación Producto GP – Factura: (tabla producto_factura) Representa los productos generadores de puntos contenidos en una factura
Atributos:

- codigo_producto_gp: clave primaria compuesta de la tabla, entero, no nulo, clave foránea hacia la tabla producto_gp, representa el producto generador de puntos contenido en la factura
- codigo_factura: clave primaria compuesta de la tabla, entero, no nulo, clave foránea hacia la tabla factura, representa la factura registrada
- cantidad: cantidad de productos del mismo tipo contenidos en la factura, entero, no nulo
- valor_unitario: valor unitario del producto generador de puntos al momento del registro, entero, no nulo

Moneda: (tabla moneda) Entidad que representa el valor de la moneda utilizada en el club
Atributos:

- codigo_moneda: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre de la moneda, varchar(20), no nulo
- valor_bs: valor en bolívares de la moneda, es decir, equivalencia a bolívares de la moneda, decimal(4, 2), no nulo
- validez_meses: tiempo de validez de la moneda en meses, entero, no nulo

Parámetro Afiliación: (tabla parametro_afiliacion) Entidad que contiene los parámetros utilizados en la generación de puntos por afiliación al club
Atributos:

- codigo_parametro: clave primaria de la tabla, entero, no nulo, aplica identidad
- tipo: tipo de afiliación (manual, con invitación, ...), varchar(15), no nulo
- cantidad_puntos: cantidad de puntos que genera el tipo de afiliación, entero, no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual del parámetro de afliación en el sistema, indica si está activo o no, entero, no nulo

Parámetro Factura: (tabla parametro_factura) Entidad que contiene los parámetros utilizados para validar las facturas registradas en el sistema
Atributos:

- codigo_parametro: clave primaria de la tabla, entero, no nulo, aplica identidad
- validez_meses: tiempo de validez de la factura en meses, entero, no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual del parámetro factura en el sistema, indica si está activo o no, entero, no nulo

Histórico Cuentas: (tabla historico_cuentas) Entidad que representa el registro de las modificaciones de las cuentas de usuario del club
Atributos:

- codigo_historico: clave primaria de la tabla, entero, no nulo, aplica identidad
- fecha_modificacion: fecha en la que la cuenta del usuario fue modificada, datetime, no nulo
- motivo: motivo por el cual la cuenta de usuario se modificó, varchar(50), nulo
- codigo_autor: clave foránea hacia la tabla usuario, representa el usuario que modificó la cuenta, entero, no nulo
- codigo_usuario: clave foránea hacia la tabla usuario, representa el usuario al cual le modificaron la cuenta, entero, no nulo
- codigo_operacion: clave foránea hacia la tabla tipo_operacion, representa la operación con la cual se modificó la cuenta, entero, no nulo

Histórico Puntos: (tabla historico_puntos) Entidad que representa el registro de las modificaciones de los estados de cuenta de cada participante
Atributos:

- codigo_historico: clave primaria de la tabla, entero, no nulo, aplica identidad
- fecha_modificación: fecha en la que el estado de cuenta del usuario fue modificado, datetime, no nulo
- tipo_puntos: tipo de puntos que fueron afectados (disponibles, redimidos, ...), varchar(20), no nulo
- cantidad_puntos: cantidad de puntos afectados, decimal(10, 2), no nulo
- codigo_autor: clave foránea hacia la tabla usuario, representa el usuario que modificó el estado de cuenta, entero, no nulo
- codigo_participante: clave foránea hacia la tabla usuario, representa el usuario al cual le modificaron el estado de cuenta, entero, no nulo
- codigo_operacion: clave foránea hacia la tabla tipo_operacion, representa la operación con la cual se modificó el estado de cuenta, entero, no nulo

Tipo Operación: (codigo_tipo_operacion) Entidad que representa las posibles operaciones sobre las cuentas de usuario y estados de cuenta de los participantes
Atributos:

- codigo_tipo_operacion: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre del tipo de operación (afiliación, canje, ...), varchar(30), no nulo
- descripcion: breve descripción explicativa del tipo de operación, varchar(50), nulo

Categoría: (tabla categoria) Entidad débil que contiene las categorías de los productos ofrecidos en el club como recompensa
Atributos:

- codigo_categoria: clave primaria de la tabla, entero, no nulo, aplica identidad
- nombre: nombre de la categoría, varchar(30), no nulo
- descripcion: breve descripción explicativa de la categoría, varchar(50), nulo
- carpeta_imagen: nombre de la carpeta donde se ubican las imágenes de las recompensas de dicha categoría, varchar(50), no nulo

Recompensa: (tabla recompensa) Entidad fuerte que representa las recompensas ofrecidas en el club

Atributos:

- codigo_recompensa: clave primaria de la tabla, entero, no nulo, aplica identidad
- codigo: código de la recompensa, varchar(10), no nulo
- nombre_corto: nombre corto de la recompensa, varchar(100), no nulo
- nombre_largo: nombre completo de la recompensa, varchar(100), no nulo
- descripcion: descripción de la recompensa, varchar(500), nulo
- precio: valor en puntos de la recompensa, decimal(10, 2), no nulo
- imagen: ruta de ubicación de la imagen de la recompensa, varchar(200), no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual de la recompensa en el sistema, entero, no nulo
- codigo_categoria: clave foránea hacia la tabla categoria, representa la categoría a la cual pertenece la recompensa, entero, no nulo

Carrito Compra: (tabla carrito_compra) Entidad que representa cada carrito de compra canjeado por los participantes

Atributos:

- codigo_carrito_compra: clave primaria de la tabla, entero, no nulo, aplica identidad
- fecha_carrito: fecha en la cual se registra el carrito de compra, datetime, no nulo
- total_compra: total en puntos del canje, decimal(10, 2), no nulo
- valor_bs_moneda: valor de la moneda en bolívares al momento del canje, decimal(4, 2), no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual del carrito de compra en el sistema, entero, no nulo
- codigo_participante: clave foránea hacia la tabla usuario, representa el usuario que registra el canje, entero, no nulo

Relación Recompensa-Carrito: (tabla recompensa_carrito) Representa los productos canjeados en un carrito de compra

Atributos:

- codigo_recompensa: clave primaria compuesta de la tabla, entero, no nulo, clave foránea hacia la tabla recompensa, representa la recompensa contenida en el carrito de compra
- codigo_carrito: clave primaria compuesta de la tabla, entero, no nulo, clave foránea hacia la tabla carrito_compra, representa el carrito de compra
- cantidad: cantidad de recompensas del mismo tipo contenidas en el carrito de compra, entero, no nulo
- valor_unitario: valor unitario en puntos de la recompensa al momento del registro del carrito de compra, decimal(10, 2), no nulo

Orden Compra: (tabla orden_compra) Entidad débil que representa la orden de compra generada por cada carrito

Atributos:

- codigo_orden_compra: clave primaria de la tabla, entero, no nulo, aplica identidad
- codigo_carrito: clave foránea hacia la tabla carrito_compra, representa el carrito de compra que genera la orden de compra, entero, no nulo
- fecha_orden: fecha en la que se registra la orden de compra, datetime, no nulo

Orden Despacho: (tabla orden_despacho) Entidad débil que representa la orden de despacho generada por cada producto en cada carrito de compra
Atributos:

- codigo_orden_despacho: clave primaria de la tabla, entero, no nulo, aplica identidad
- codigo_compra: clave foránea hacia la tabla orden_compra, representa la orden de compra que genera la orden de despacho, entero, no nulo
- fecha_orden: fecha en la que se registra la orden de despacho, datetime, no nulo
- codigo_estatus: clave foránea hacia la tabla estatus, representa el estatus actual de la orden de despacho, entero, no nulo
- direccion_envio: dirección a la que va a ser enviada la recompensa, varchar(400), no nulo

Dirección: (tabla direccion) Entidad débil que representa la dirección de un usuario o sucursal
Atributos:

- codigo_direccion: clave primaria de la tabla, entero, no nulo, aplica identidad
- direccion_completa: dirección completa del usuario y/o sucursal, varchar(400), no nulo

CONCLUSIONES

El objetivo general de este Trabajo Especial de Grado fue logrado a cabalidad cumpliendo con los requerimientos de la aplicación web de programa de incentivos acordados en esta investigación así como con los tiempos de entrega.

El método de desarrollo utilizado comprendió la división del proyecto en cuatro (fases) según la metodología RUP: inicio, elaboración, construcción y transición. Esto aumentó el rendimiento de los tiempos de trabajo y le dio el formalismo suficiente al proyecto puesto que se tomó de la metodología pesada RUP la entrega de documentos y diagramas, realmente necesarios para el desarrollo del proyecto, en la fase de inicio y de elaboración: diagramas de casos de uso, descripción de los mismos, diagrama de clases, diagramas de secuencia del sistema, de colaboración, de actividades y diagramas de navegabilidad (WAE); y en la fase de transición se realizó un diccionario de datos que sirve de apoyo para futuros mantenimientos de la base de datos. Las recomendaciones de la metodología ágil Scrum se tomaron en cuenta para la fase de construcción realizando semanalmente entregables funcionales, dividiendo el trabajo de tal modo que cada entregable de software fuese funcionalmente superior al anterior y, realizando las pruebas del software en paralelo a la construcción de otros módulos del sistema.

El análisis necesario para la puesta en marcha de la implementación del sistema fue realizado en las fases de inicio y elaboración logrando conocer con un alto nivel de detalle la forma de ejecución de cada uno de los flujos dentro de un programa de incentivos y reflejándolos en cada uno de los diagramas realizados. Además en dichas fases también se realizaron los diagramas entidad-relación para la posterior creación de la base de datos necesaria para el almacenamiento de toda la información necesaria para la aplicación de programa de incentivos.

La implementación fue dividida según los módulos necesarios en la aplicación: participación en el programa de incentivos, administración del mismo y administración de la tienda. Éstos fueron subdivididos para cumplir con la premisa de Scrum (“un incremento funcional en cada iteración”). El primer módulo, de participación, se subdividió en afiliación al programa de incentivos, registro de compras y canje de recompensas. El segundo módulo y tercer módulo, de administración del programa de incentivos y de la tienda, respectivamente, se subdividió en la administración de los elementos (sucursales, productos, recompensas, etc.) y en el sub-modulo de reportes.

Este Trabajo Especial de Grado sugiere el desarrollo de un Generador de Programas de Incentivos, pudiendo ser éste tomado como base para el estudio y desarrollo de una aplicación que mediante la configuración de parámetros y reglas de negocios predefinidas y comunes a la mayoría de los programas de lealtad permita generar de forma sencilla programas de incentivos personalizados según la empresa que lo requiera. También se sugiere para la realización del generador utilizar la metodología de desarrollo de software utilizada en este Trabajo Especial de Grado, la cual a groso modo es una combinación entre la metodología RUP y la metodología Scrum, ya que como bien se ha mencionado anteriormente permite rendir en el desarrollo del proyecto disminuyendo los tiempos de entrega y formaliza el proyecto con la documentación.

BIBLIOGRAFÍA

Cárdenas, L., & Gracia, J. (s.f.). *Web Estlio: Conceptos Básicos*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Web Estilo: http://www.webestilo.com/javascript/js00.phtml

Castillo, C. (s.f.). *Artículos: Lenguajes de Programación para la Web*. Recuperado el 6 de Agosto de 2008, de sitio Web de Tejedores del Web: http://www.tejedoresdelweb.com/307/article-1883.html

De La Torre, A. (2006). *Documentos: Lenguaje del Lado del Servidor o Cliente*. Recuperado el 6 de Agosto de 2008, de sitio Web de Adelat: http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

Dichiara, T. (2 de Marzo de 2004). *Artículos: Hacerle caso a SQL*. Recuperado el 8 de Agosto de 2008, de sitio Web de Tech: http://searchsqlserver.techtarget.com/news/article/0,289142,sid87_gci1124761,00.html

Dimagin. (2007). *Dimagin: Las Aplicaciones Web*. Recuperado el 2008 de Septiembre de 11, de sitio Web de Dimagin: http://www.dimagin.net/es/contenido.php?t id=6

Eguiluz Pérez, J. (2008). *Introducción a CSS: ¿Qué es CSS?* Recuperado el 4 de Septiembre de 2008, de sitio Web de Libros Web: http://www.librosweb.es/css/capitulo1.html

Eguiluz Pérez, J. (4 de Agosto de 2008). *Introducción a JavaScript: ¿Qué es JavaScript?* Recuperado el 4 de Septiembre de 2008, de sitio Web de Libros Web: http://www.librosweb.es/javascript/capitulo1.html

Ercoli, J. (30 de Noviembre de 2007). *Arquitectura ASP Net Clásica: Modelo de WebForms*. Recuperado el 18 de Noviembre de 2008, de sitio Web de BlogSpot: http://metodogiasdesistemas.blogspot.com/2007/11/arquitectura-asp-net-clsica-modelo-de.html

Gómez Gallego, J. P. (16 de Septiembre de 2007). *Documentos: Fundamentos de la Metodología RUP Rational Unified Process*. Recuperado el 31 de Julio de 2008, de sitio Web de Scribd: http://www.scribd.com/doc/297224/RUP

González Estrada, J. A. *Módulo: Aplicaciones para Internet en .NET.*

Gonzalez Urmachea, M. (s.f.). *Monografías: SQL Server*. Recuperado el 18 de Noviembre de 2008, de sitio Web de Monografías: http://www.monografias.com/trabajos14/sqlserver/sqlserver.shtml

González, G., & Machuca, M. (2008). *Archivos: Tecnologías Web Modelo Vista Controlador (MVC).* Recuperado el 8 de Agosto de 2008, de sitio Web de Open Solutions: http://www.opensolutions.com.py/~ggonzalez/fpuna/is2/files/06_TECNOLOGIAS_WEB_MVC.pdf

Good, R. (2 de Noviembre de 2005). *Artículos: Beneficios de las Aplicaciones Basadas en Web y el Anuncio de Microsoft de la Era "En Vivo"*. Recuperado el 2008 de Septiembre de 11, de sitio Web de Master New Media: http://www.masternewmedia.org/es/aplicaciones_web/temas_de_aplicaciones_web/Beneficios_De_Las_Aplicaciones_Basadas_En%20_Web_Y_El_Anuncio_De_Microsoft_De_La_Era_En_Vivo.htm

Gracia, J. (4 de Septiembre de 2006). *Equipos: Gestión de Proyectos con Scrum*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Ingenieros de Software: http://www.ingenierosoftware.com/equipos/scrum.php

Gutiérrez Mayoral, A. (11 de Febrero de 2005). *GSYC: Patrón de Diseño MVC*. Recuperado el 3 de Octubre de 2008, de sitio Web de GSYC: http://gsyc.es/~agutierr/pfc-tecnica-html/node43.html

Gúzman Matus, O. E. (s.f.). *OEGuzman: SCRUM Metodología Ágil para la Planificación y Seguimiento de Proyectos*. Recuperado el 4 de Septiembre de 2008, de sitio Web de GooglePages: http://oeguzman.googlepages.com/

Janium. (2007). *Janium: Aplicaciones Basadas en Web*. Recuperado el 11 de Septiembre de 2008, de sitio Web de Janium: http://www.janium.com/page2/page1/page6/page7/page7.html

Kioskea. (s.f.). *Kioskea: Entorno Cliente/Servidor*. Recuperado el 16 de Septiembre de 2008, de sitio Web de Kioskea: http://es.kioskea.net/cs/csintro.php3

Loja, F. (Septiembre de 2007). *Archivos: Construcción de Aplicaciones en Capas.* Recuperado el 6 de Agosto de 2008, de sitio Web de Fausto Loja: http://faustol.files.wordpress.com/2007/09/construccion-de-aplicaciones-en-capas.doc

Manuales PDF. (2007). *Manuales PDF: Manual de ASP.NET.* Recuperado el 8 de Agosto de 2008, de sitio Web de Manuales PDF: http://www.manualespdf.es/manual-asp-net/

Martínez Echeverría, Á. (1995). *Manuales: Manual Práctico de HTML*. Recuperado el 4 de Septiembre de 2008, de sitio Web de UPM: http://www-app.etsit.upm.es/~alvaro/manual/manual.html

Microsoft Corporation. (2001). *GeoCities: Tutorial de ASP.NET*. Recuperado el 31 de Junio de 2008, de sitio Web de GeoCities: http://es.geocities.com/rocadanny74/hwct/T3/quickstart.htm

Microsoft. (2008). *Microsoft: ¿Qué es SQL Server?* Recuperado el 18 de Noviembre de 2008, de sitio Web de Microsoft: http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx

Papyros Digitales. (s.f.). *Artículos: Las Tecnologías de las Páginas Web*. Recuperado el 6 de Agosto de 2008, de sitio Web de Papyros Digitales: http://www.webtaller.com/maletin/articulos/las_tecnologias_de_las_paginas_web.php

Programación Web. (13 de Abril de 2007). *Artículos: MVC - Modelo Vista Controlador*. Recuperado el 19 de Octubre de 2008, de sitio Web de Programación Web: http://www.programacionweb.net/articulos/articulo/?num=505

Ravioli, P. (s.f.). *Monografás: Lenguaje de Programación para Páginas Web HTML*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Monografías: http://www.monografias.com/trabajos7/html/html.shtml

Rodríguez Terrero, P. N. (28 de Julio de 2005). *Aplicaciones Distribuidas de 3 Capas: Parte I/IV*. Recuperado el 6 de Agosto de 2008, de sitio Web de El Guille: http://www.elguille.info/colabora/NET2005/Sagara_AplicacionesDistribuidas3Capas.htm

Sánchez Gónzalez, C. (28 de Septiembre de 2004). *ONess: Un Proyecto Open Source para el Negocio Textil Mayorista Desarrollado con Tecnologías Open Source Innovadoras*. Recuperado el 6 de Agosto de 2008, de sitio Web de ONess: http://oness.sourceforge.net/proyecto/html/ch03s02.html

Serrano, J. (9 de Mayo de 2007). *Archivos: Explicando Scrum a mi Abuela*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Geeks: http://geeks.ms/blogs/jorge/archive/2007/05/09/explicando-scrum-a-mi-abuela.aspx

Silice. (s.f.). *Escenarios Globales: Características Físicas*. Recuperado el 16 de Septiembre de 2008, de sitio Web de Silice: http://www.csi.map.es/csi/silice/Escglo5.html

Silice. (s.f.). *Escenarios Globales: Ventajas e Inconvenientes*. Recuperado el 16 de Septiembre de 2008, de sitio Web de Silice: http://www.csi.map.es/csi/silice/Global75.html

Valle, J. G., & Gutiérrez, J. G. (2005). *Monografías: Definición Arquitectura Ciente Servidor*. Recuperado el 16 de Septiembre de 2008, de sitio Web de Monografías: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

Vegas, J. (21 de Marzo de 2002). *Módulos: Introducción a las Aplicaciones Web*. Recuperado el 11 de Septiembre de 2008, de sitio Web de Cursos de Jesús Vegas: http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html

Wikipedia. (6 de Septiembre de 2008). *Wikipedia: Aplicación Web*. Recuperado el 11 de Septiembre de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

Wikipedia. (10 de Septiembre de 2008). *Wikipedia: Cliente-Servidor*. Recuperado el 16 de Septiembre de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/Cliente-Servidor

Wikipedia. (2 de Septiembre de 2008). *Wikipedia: Hojas de Estilo en Cascada*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/CSS

Wikipedia. (29 de Agosto de 2008). *Wikipedia: HTML*. Recuperado el 4 de Septiembre de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/Codigo_HTML

Wikipedia. (28 de Septiembre de 2008). *Wikipedia: Modelo Vista Controlador*. Recuperado el 3 de Octubre de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Wikipedia. (17 de Julio de 2008). *Wikipedia: Proceso Unificado de Rational*. Recuperado el 31 de Julio de 2008, de sitio Web de Wikipedia: http://es.wikipedia.org/wiki/RUP

ANEXOS

Diagramas de colaboración

Creación de Producto Generador de Puntos (GP)

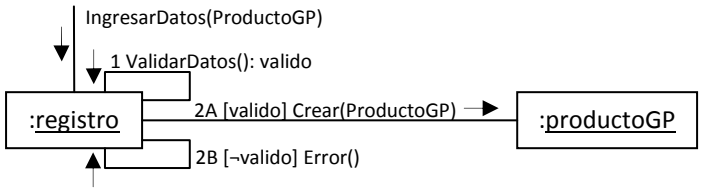

Figura 1.A. Diagrama de colaboración "Creación de Producto GP"

Consulta de Producto Generador de Puntos (GP)

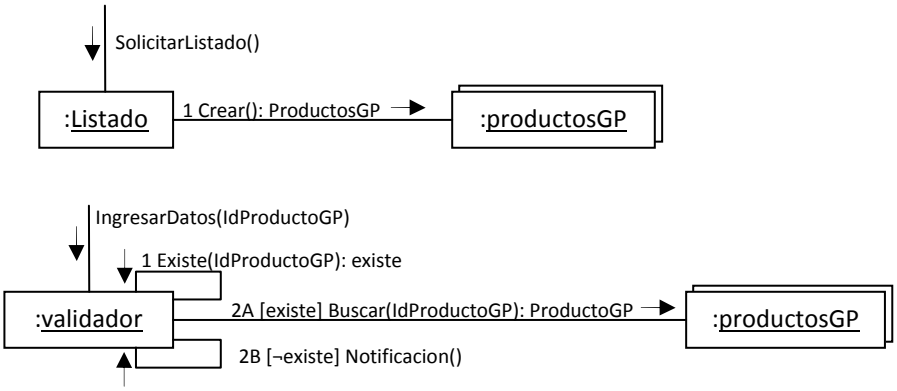

Figura 2.A. Diagrama de colaboración "Consulta de Producto GP"

Modificación de Producto Generador de Puntos (GP)

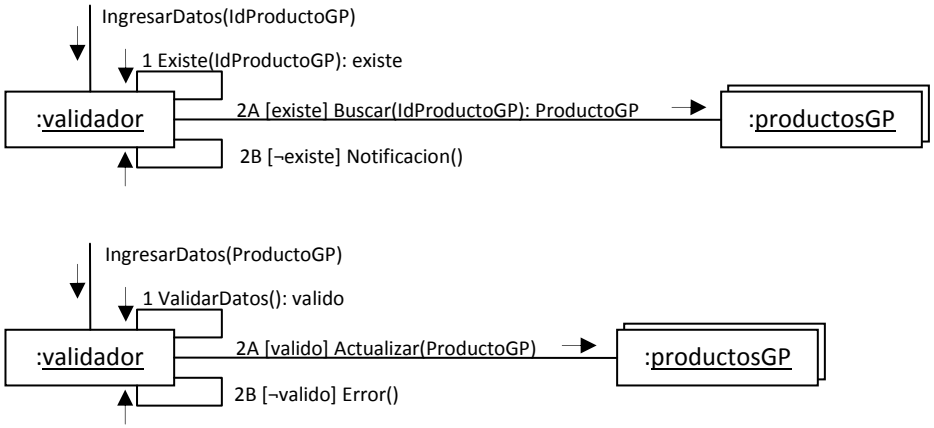

Figura 3.A. Diagrama de colaboración "Modificación de Producto GP"

Consulta de Cuenta

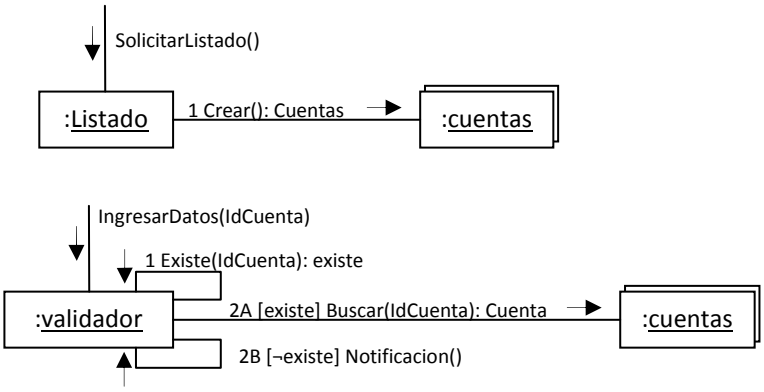

Figura 4.A. Diagrama de colaboración "Consulta de Cuenta"

Modificación de (Mi) Cuenta

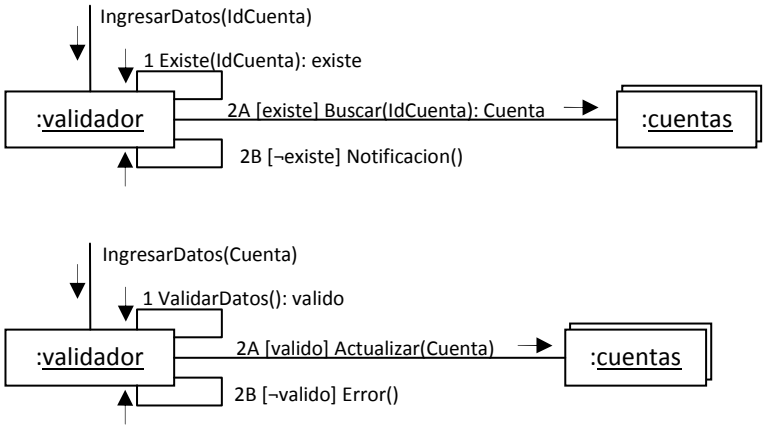

Figura 5.A. Diagrama de colaboración "Modificación de (Mi) Cuenta"

Consulta de Orden de Despacho (OD)

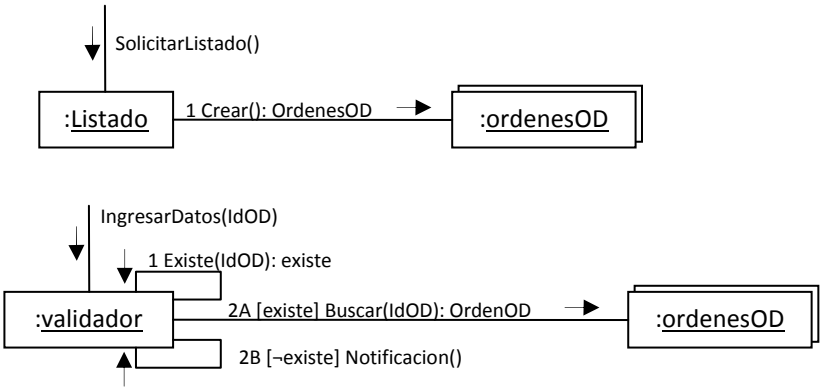

Figura 6.A. Diagrama de colaboración "Consulta de Orden de Despacho"

Modificación del Estatus de Orden de Despacho (OD)

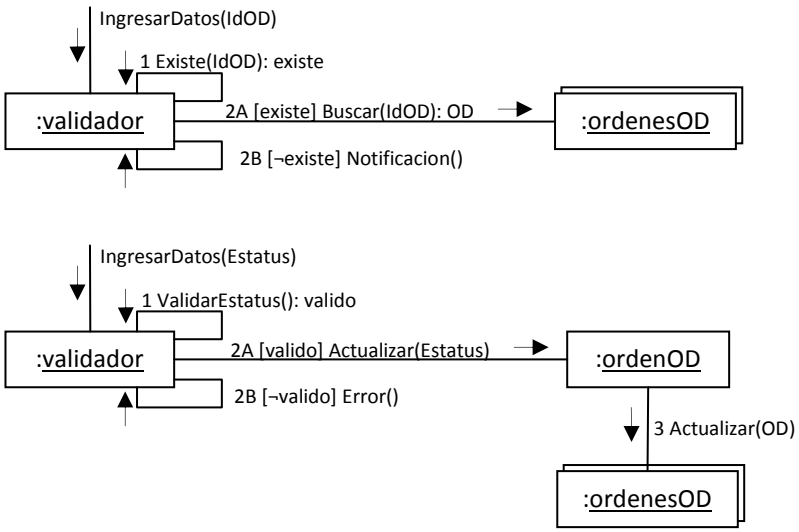

Figura 7.A. Diagrama de colaboración "Modificación del Estatus de Orden de Despacho"

Consulta de Mi Cuenta

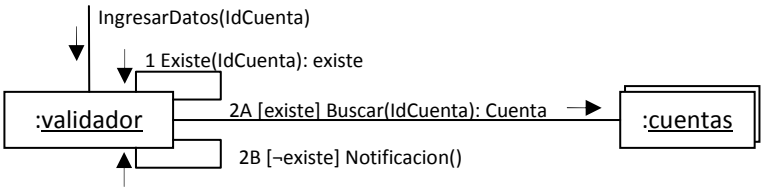

Figura 8.A. Diagrama de colaboración "Consulta de Mi Cuenta"

Diagramas de secuencia del sistema

Administrar ← Administrar Producto GP – Crear Producto GP

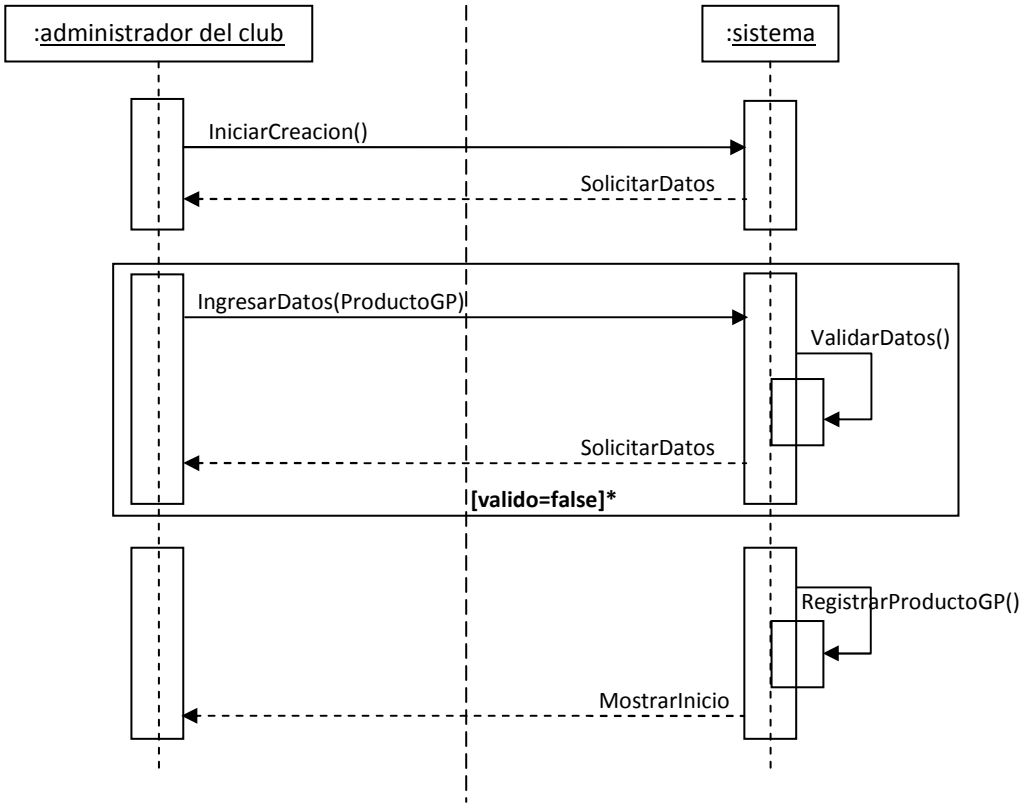

Figura 9.A. Diagrama de secuencia “Crear Producto GP”

Administrar ← Administrar Producto GP – Consultar Producto GP

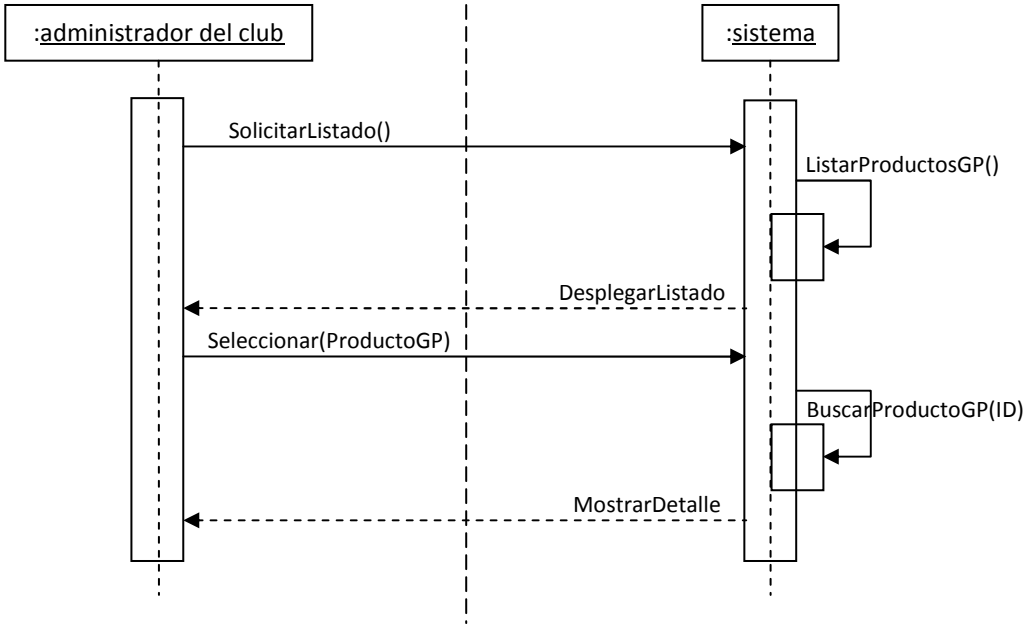

Figura 10.A. Diagrama de secuencia "Consultar Producto GP"

Administrar ← Administrar Producto GP – Modificar Producto GP

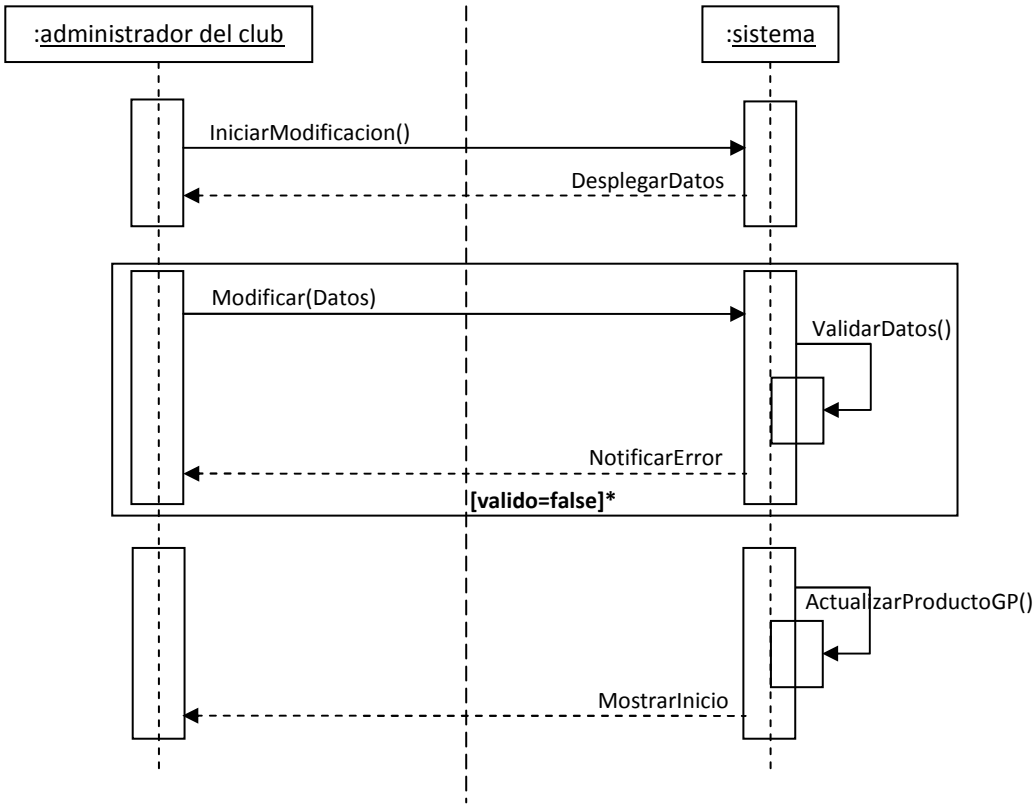

Figura 11.A. Diagrama de secuencia "Modificar Producto GP"

Administrar ← Administrar Cuenta – Consultar Cuenta

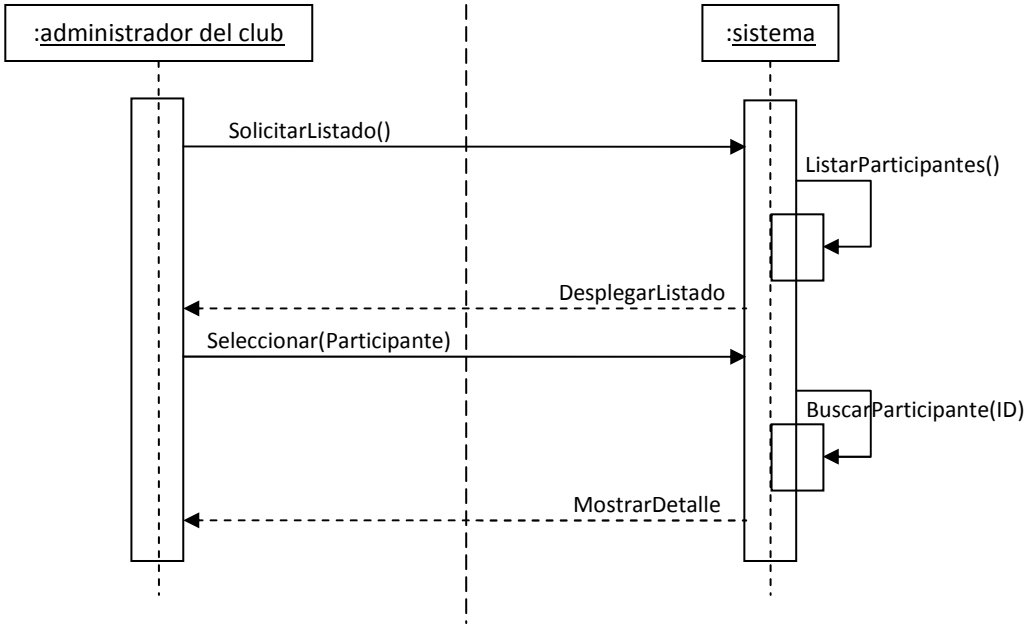

Figura 12.A. Diagrama de secuencia "Consultar Cuenta"

Administrar ← Administrar (Mi) Cuenta – Modificar (Mi) Cuenta

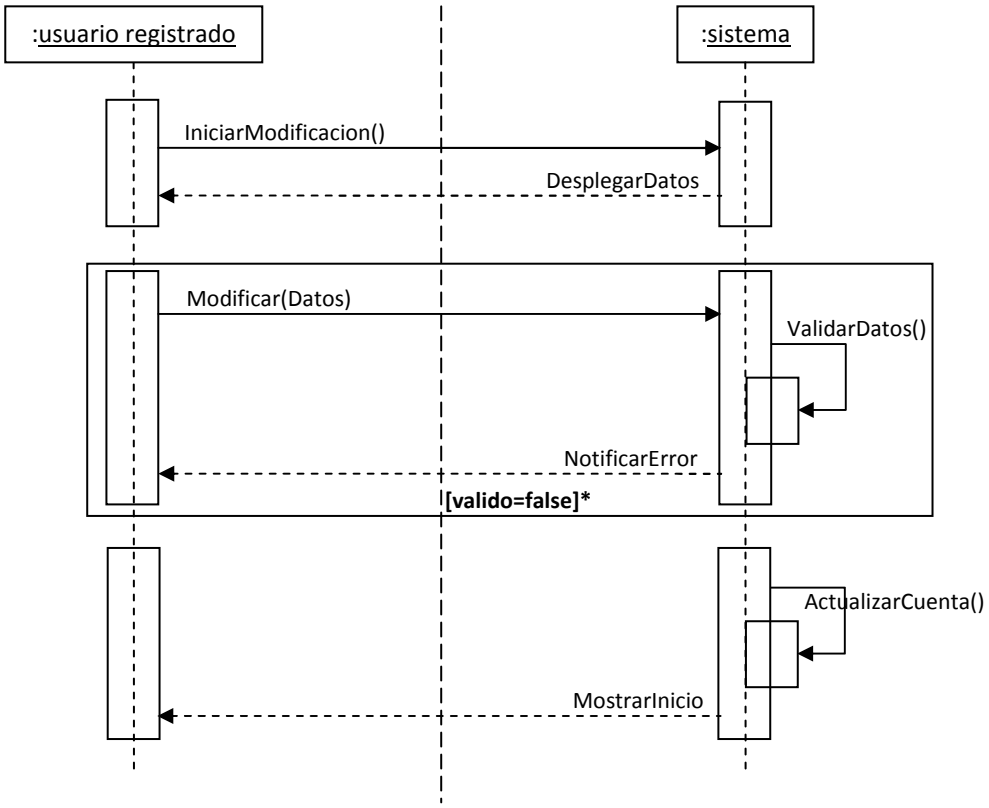

Figura 13.A. Diagrama de secuencia "Modificar (Mi) Cuenta"

Administrar ← Administrar Orden de Despacho (OrDes) – Consultar Orden de Despacho

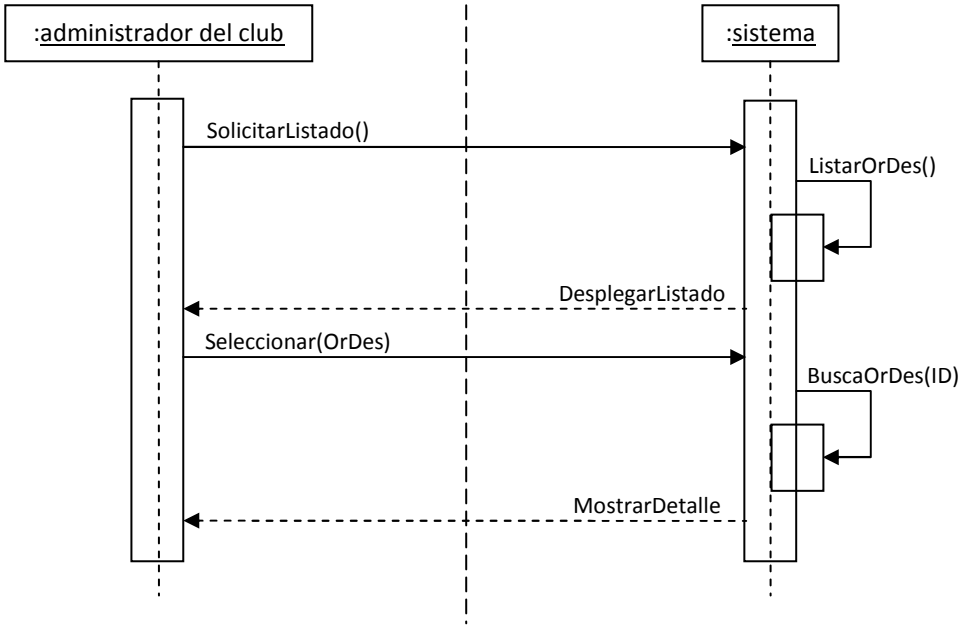

Figura 14.A. Diagrama de secuencia "Consultar Orden de Despacho"

Administrar ← Administrar Orden de Despacho (OrDes) – Modificar Estatus de Orden de Despacho (OrDes)

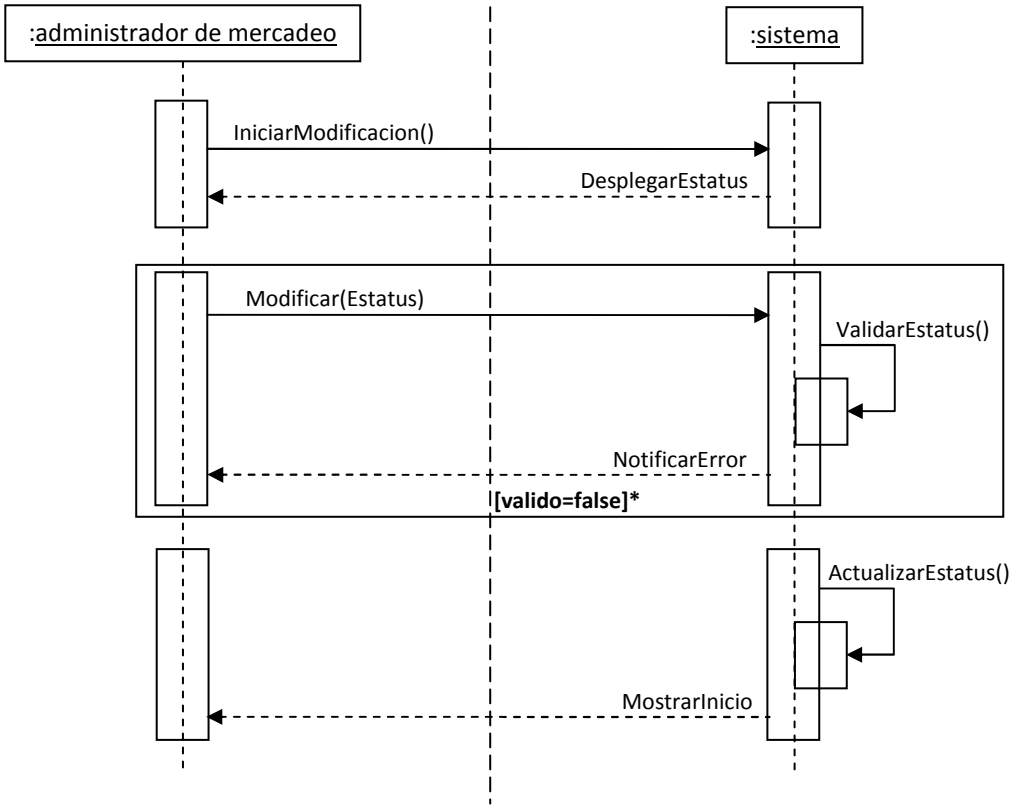

Figura 15.A. Diagrama de secuencia "Modificar Estatus de Orden de Despacho"

Administrar Mi cuenta – Consultar Mi Cuenta

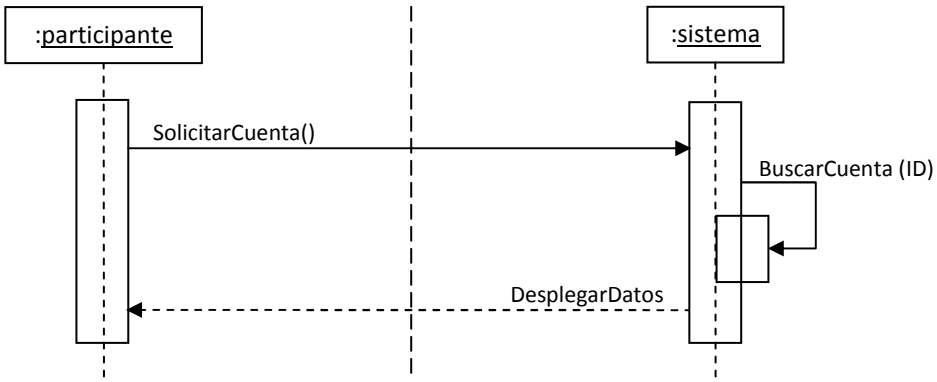

Figura 16.A. Diagrama de secuencia “Consultar Mi Cuenta”

Diagramas de actividades

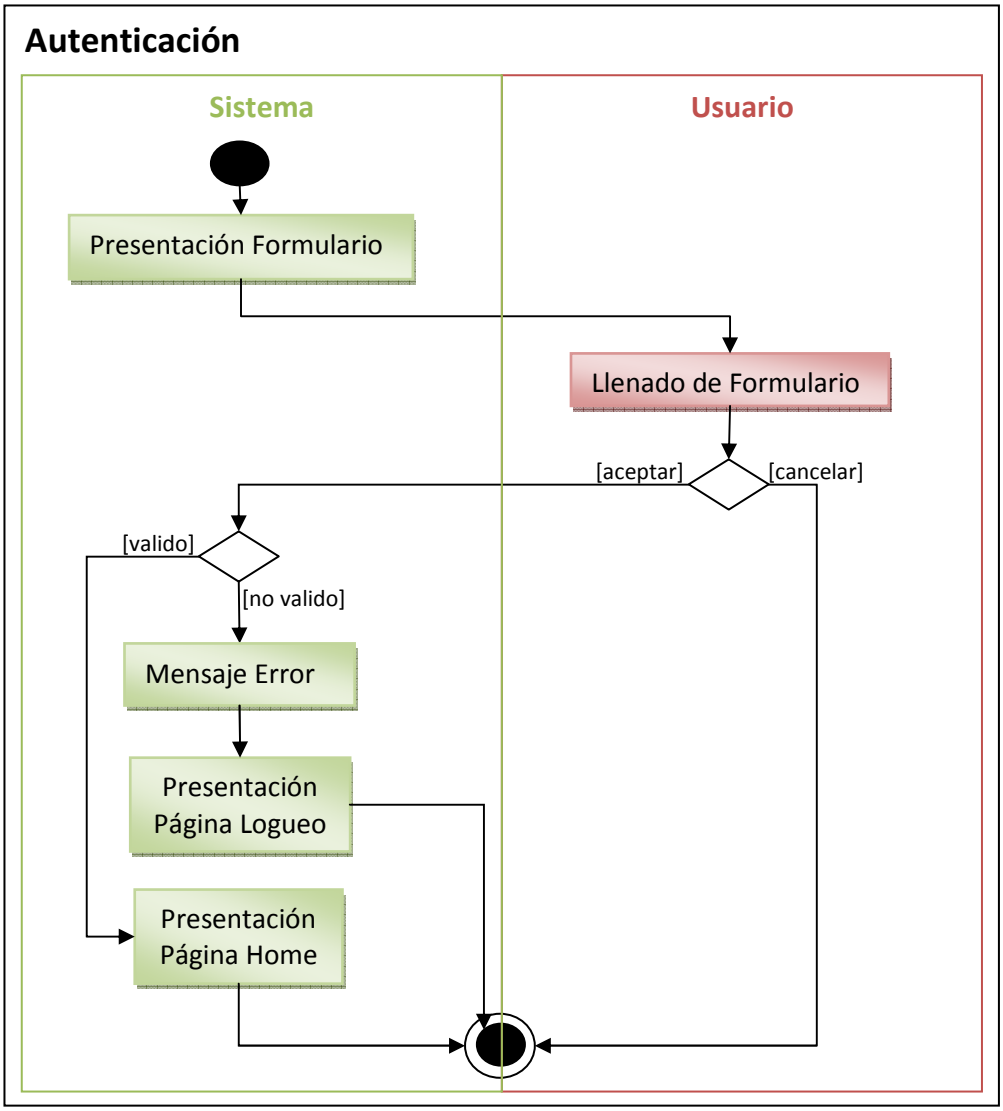

Figura 17.A. Diagrama de actividades “Autenticación”

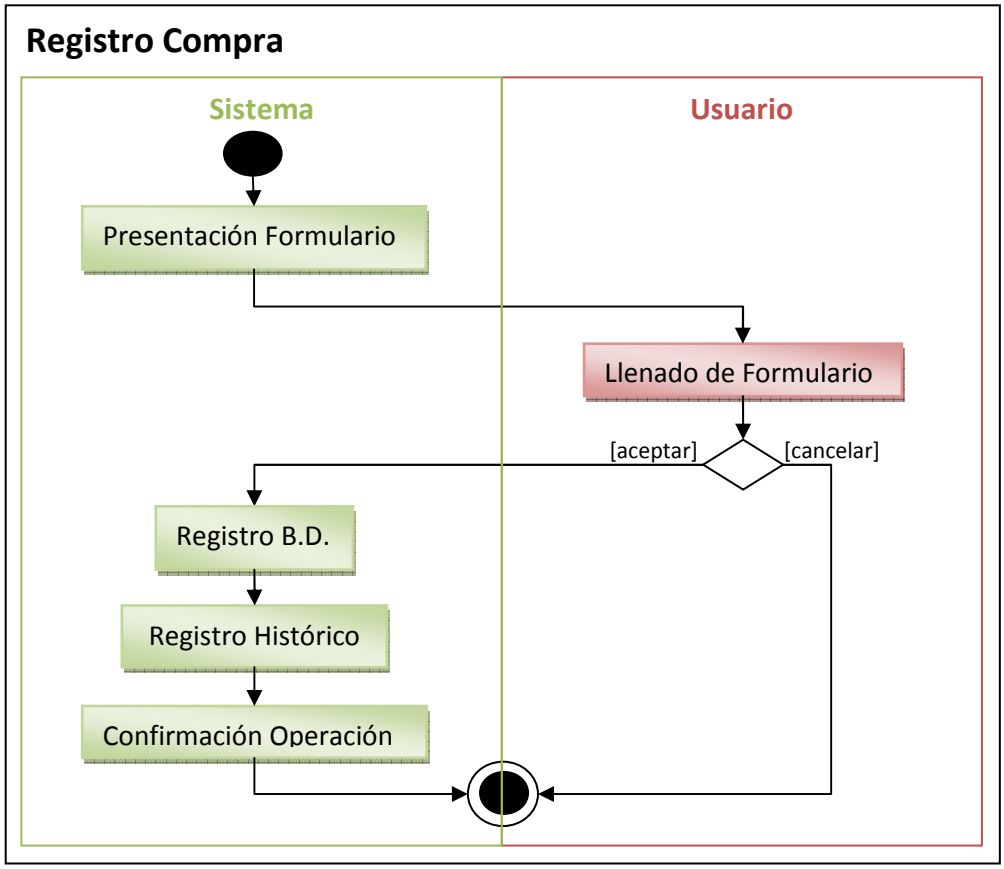

Figura 18.A. Diagrama de actividades “Registro de Compra”

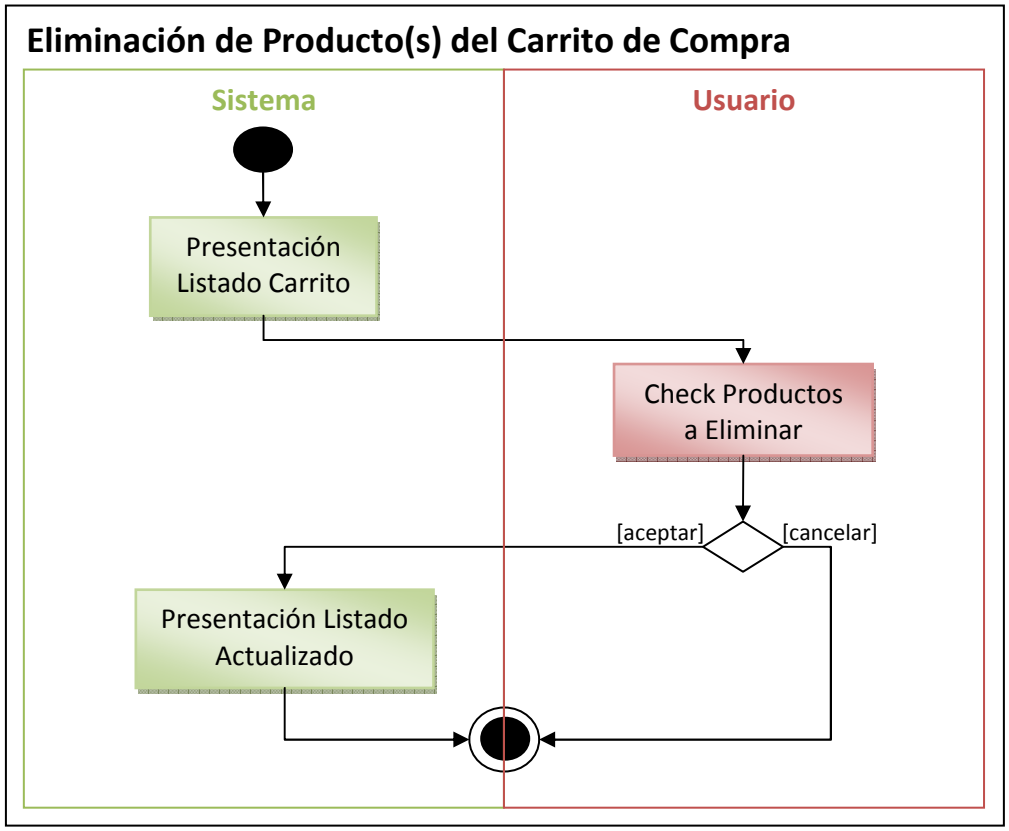

Figura 19.A. Diagrama de actividades “Eliminación de Producto(s) del Carrito de Compra”

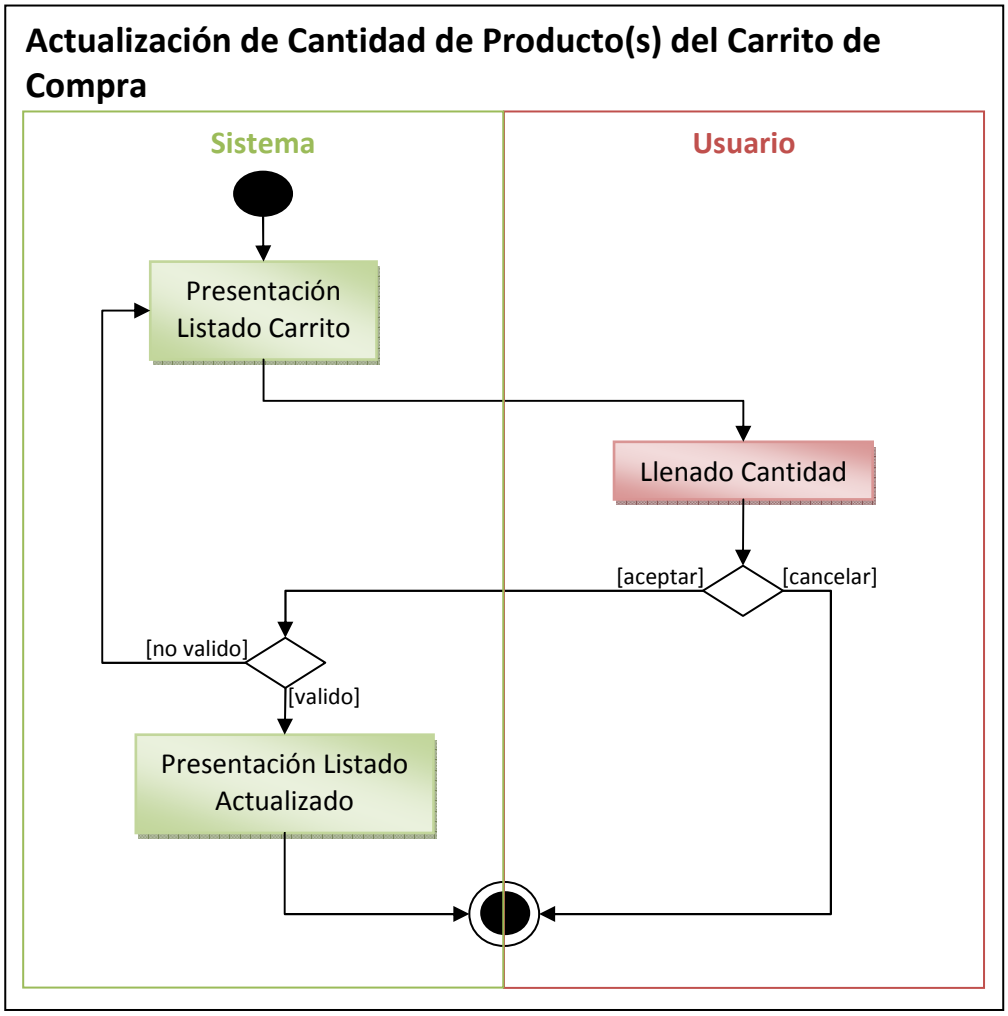

Figura 20.A. Diagrama de actividades “Actualización de Cantidad de Producto(s) del Carrito de Compra”

DIAGRAMA NAVEGACIÓN SECUNDARIA I

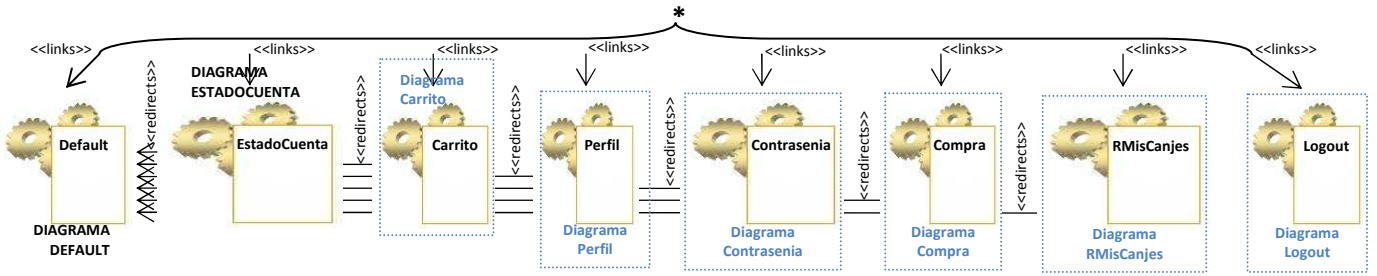

DIAGRAMA NAVEGACIÓN SECUNDARIA II

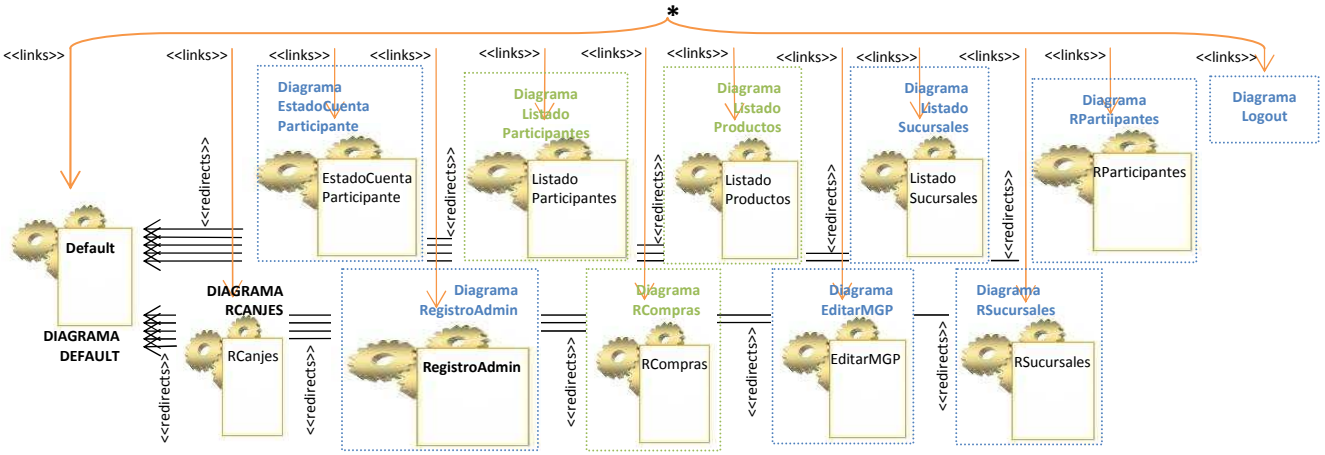

DIAGRAMA NAVEGACIÓN SECUNDARIA III

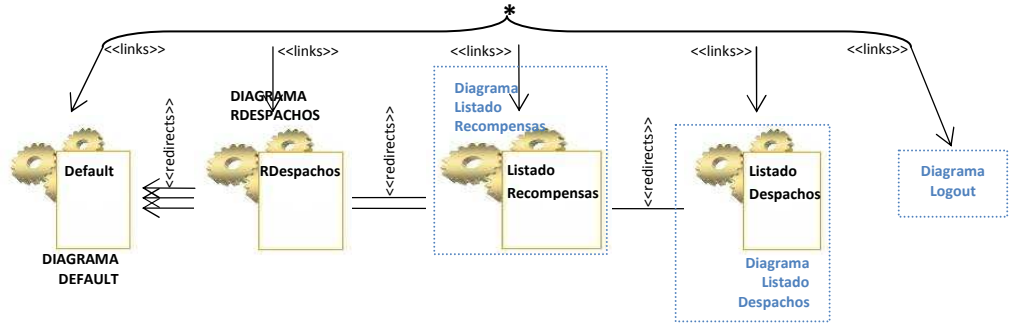

Figura 21.A. Diagrama WAE “Navegación Secundaria”

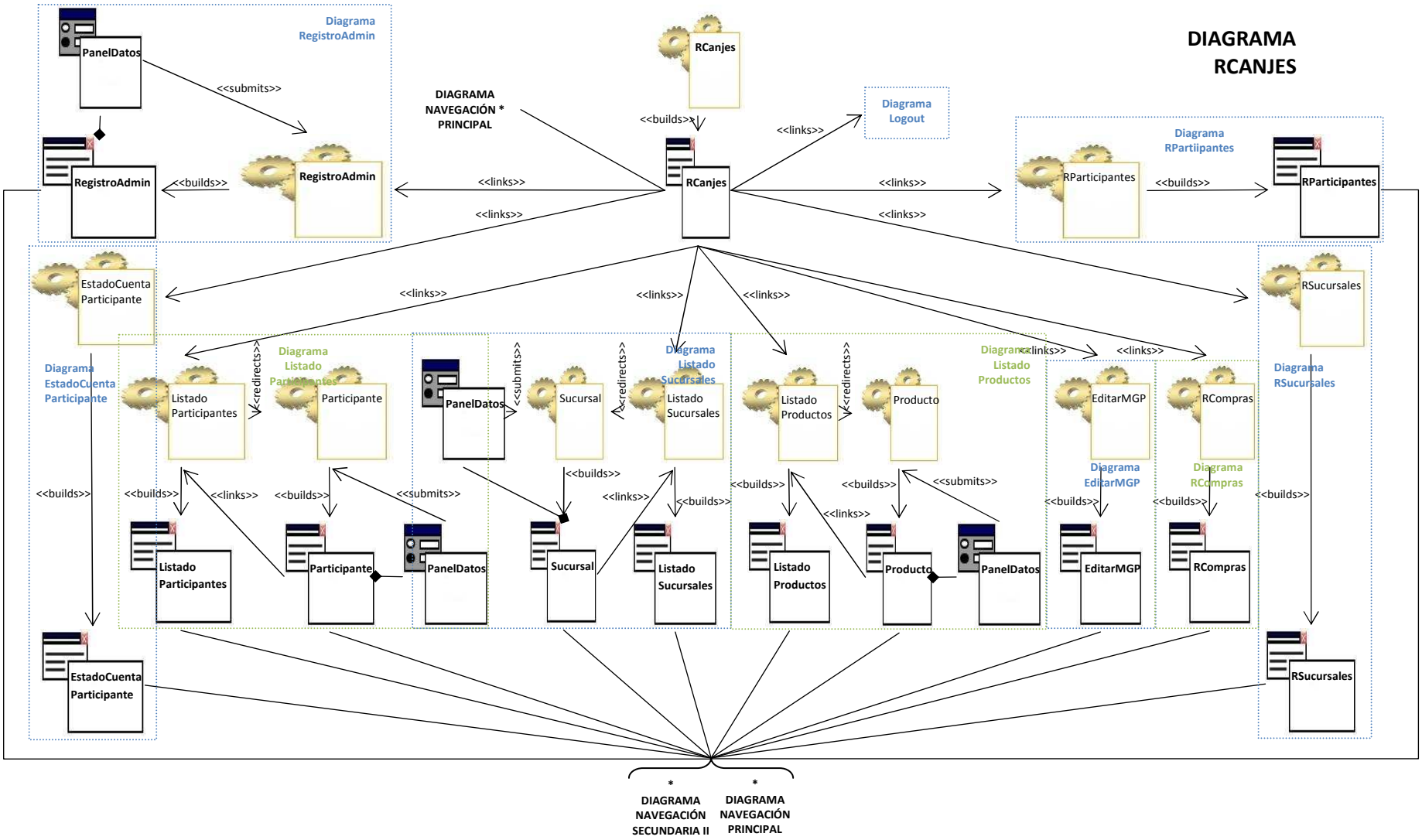

Figura 22.A. Diagrama WAE “Reporte de Canjes”

DIAGRAMA RDESPACHOS

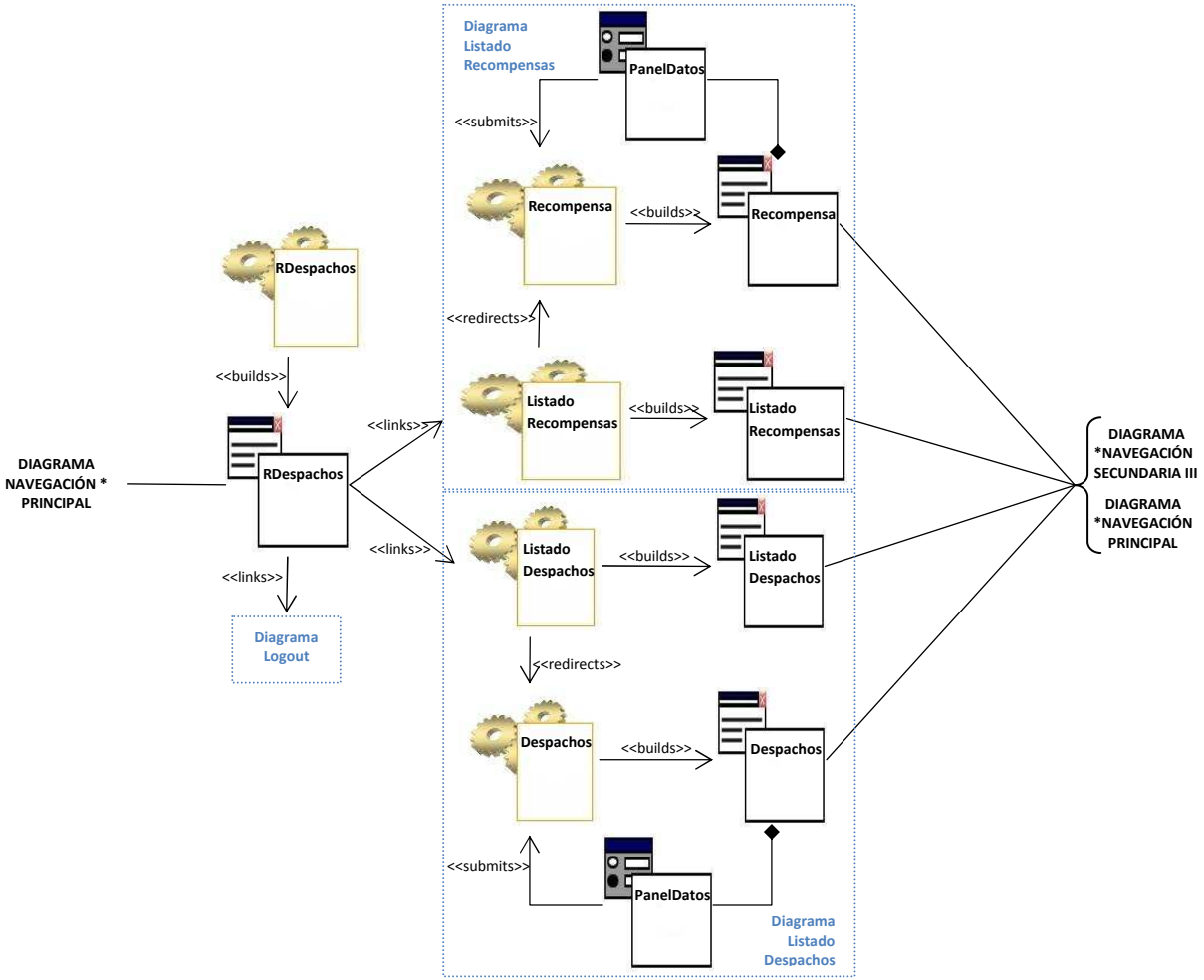

Figura 23.A. Diagrama WAE “Reporte de Órdenes de Despacho”