Universidad Del Azuay

Facultad De Ciencias De La Administración

Escuela De Ingeniería De Sistemas

"Aplicación práctica de RIA (Rich Internet Applications) en el comercio electrónico con el desarrollo de un Sistema de Entregas y Repartos DELIVERY"

Tesis previa a la obtención del título de

Ingeniero De Sistemas

AUTORES:

Daysi Marlene Becerra Naranjo

Juan Andrés Vélez Zea

DIRECTOR DE TESIS:

Ing. Pablo Pintado Zumba

CUENCA, ECUADOR 2011

Dedicatoria

La presente tesis está dedicada a mi familia, por el gran apoyo y cariño que he tenido de cada uno de ellos. A mi esposo Juan Andrés quien siempre me ha demostrado su gran amor, su paciencia y por haber estado todo este tiempo a mi lado para juntos apoyarnos en la culminación de esta tesis; ya que él es y será siempre mi inspiración en todo proyecto que pueda emprender. A mi querida madre Aida, quien siempre ha velado por mi bienestar y educación en todo momento, y me motivó siempre para alcanzar mis metas confiando en mí en todo momento. A mi querido padre Efraín, quien supo educarme y darme su amor. A mi querida abuelita Enriqueta quién siempre está a nuestro lado para brindarnos todo su apoyo y cariño; a mis queridas hermanas Daniela y Tatiana y mi sobrinita Sophia, quienes siempre me acompañaron y estuvieron a mi lado siempre.

Daysí Becerra Naranjo

Mi trabajo de tesis está dedicado a mi esposa Daysí por ser ella quien inspira mis sueños y me alienta para hacerlos realidad, para mi padre Raúl por enseñarme el valor de la humildad y la alegría del trabajo bien realizado, a mi madre Patricia por enseñarme el valor de creer en uno mismo, a mis hermanos Francisco y Raúl por confiar en mí y recordarme que nunca estoy solo.

Es un placer dedicar este trabajo a mis tíos Iván Torres y Cecilia Zea y a mi abuelita Rebeca por ser ellos quienes han apoyado y seguido mi andar a lo largo de toda mi vida, compartiendo conmigo lo mejor de sí, lo que me ha permitido tener grandes ejemplos a imitar.

A mis familiares y amigos de la universidad y de la infancia dedico también este trabajo.

Juan Andrés Vélez Zea

Agradecimientos

Agradezco en primer lugar a Jehová Dios por ser mi guía y mi fortaleza en todos los pasos de mi vida. A Juan Andrés, mi querido esposo, por estar siempre a mi lado y trabajar conmigo todo este tiempo para juntos lograr este gran sueño.

A mi querida Madre, ella siempre cuida de mí y gracias a ella hoy puedo tener una carrera profesional para mi futuro. A mi muy querida abuelita Enriqueta y Tío Walter Naranjo por apoyarme siempre en mis estudios y darme su apoyo incondicional.

Un sincero agradecimiento para nuestro director de tesis Ing. Pablo Pintado, quién con su paciencia y gran conocimiento nos ha sabido guiar a lo largo del desarrollo de esta tesis y sobre todo por su gran calidad humana, por demostrarnos que lo primero para llegar a ser un gran profesional es ser un gran ser humano. Agradezco a todos aquellos quienes confiaron en mí, a mis compañeros, amigos y profesores, ya que sin ellos no hubiese sido posible este sueño.

Daysi Becerra Naranjo

Agradezco primeramente a Jehová mi Dios por permitirme recorrer el camino que me he trazado siendo mi roca fuerte en momentos de adversidad.

A mi amada esposa por apoyar tan diligentemente mi trabajo y cuidar tan gentilmente de mi bienestar pidiendo nada a cambio y brindando diariamente un cálido e inspirador beso. A mis padres Raúl y Patricia por creer siempre en el potencial de mis ideas y siendo refugio inagotable de amor, paciencia, bondad y corrección. Agradezco de manera especial al Ingeniero Pablo Pintado por su guía y generosidad al compartir sus conocimientos que de gran manera aportaron al desarrollo y culminación de esta Tesis de Grado.

Juan Andrés Vélez Zea

Índice De Contenidos

Índice De Ilustraciones

Capítulo I

Capítulo II

Capítulo III

Capítulo IV

Capítulo VI

Capítulo VII

Índice De Tablas

Índice De Anexos

Resumen

La presente tesis muestra las nuevas tendencias de programación de Sitios Web usando tecnología RIA, la cual agrega mayor potencial a estas aplicaciones sobrepasando las barreras de navegador, hardware, Sistema Operativo y Gadgets que han sido un obstáculo para los desarrolladores.

Se ejemplificó esto mediante la creación de una aplicación llamada QuickDelivery, para comercio electrónico, usando Adobe Flex como Framework y ColdFusion como servidor para consumir servicios Web, a la misma que agregamos la funcionalidad de las API´s de Google Maps y Languaje. Nuestra aplicación funciona tanto OnLine como OffLine gracias al uso de Adobe Air que nos permite tener la aplicación en el navegador y en el escritorio del usuario.

ABSTRACT

The present Thesis shows the tendencies of Web Site programming with the use of RIA technology. This technology adds more potential to these applications and surpasses the barriers of the navigator, the hardware, the Operative System and the Gadgets, which have been an obstacle for the developers.

An application known as Quick Delivery was applied for electronic commerce purposes, using Adobe Flex as a Framework and Cold Fusion as a server for Web consuming services. We added a Google Map and Language API. Thanks to the use of Adobe Air, we were able to include the application in the navigator and on the user's desk. This will make it possible to work On Line as well as Off Line.

Translated by,

Diana Lee Rodas

INTRODUCCIÓN

En la actualidad todos hemos experimentado los beneficios que nos provee el uso del Internet tanto con fines académicos, comerciales, sociales y de comunicación en general; llegando a convertirse más que en un elemento de entretenimiento en una herramienta de trabajo y desarrollo de los pueblos alrededor del mundo. En la actualidad el número de personas que accede al Internet crece de manera exponencial cada día, y también sus exigencias sobre los servicios y beneficios que sobre este pueden obtener, y es así como los actuales navegadores se encuentran cada vez más exigidos frente al desarrollo de Internet y a la enorme cantidad de aplicaciones que se han sumado a la red en los últimos años. Música en línea, la posibilidad de cargar imágenes y videos en blogs y espacios de redes sociales, sitios de almacenamiento virtual, páginas con presentaciones multimedia, documentos de diferente tipo que se editan online en forma compartida, como en el caso de Google Docs, todas estas aplicaciones tornan más "pesada" y compleja a la web y, en consecuencia, ponen a los navegadores actuales en el límite de sus posibilidades.

Ante esto, han surgido herramientas web que buscan concretar un paso adelante tecnológicamente hablando, lo que permitiría una actividad más provechosa en Internet. Esta tecnología se denomina RIA (Rich Internet Applications) y permiten descargar en un único paso toda la información necesaria, sin el intercambio con los servidores que caracteriza a la tecnología actual, además permite el uso de los recursos propios del sistema sin tener que depender únicamente del navegador. En consecuencia, la navegación y las tareas en la web se vuelven más fluidas.

Para llevar a la práctica lo antes descrito con la tecnología RIA, hemos pensado en el desarrollo de un sistema de comercio electrónico al que llamaremos Quick Delivery, el cual se preocupa del diseño, planificación,

implementación y mejoramiento de los flujos asociados a la entrega de productos comestibles.

Para llevar a cabo estos objetivos vemos la necesidad de aprender a usar nuevas herramientas que nos den la capacidad de crear aplicaciones bajo los estándares de la tecnología RIA, como es la aplicación Adobe Flex Builder 3, la misma que usaremos para desarrollar nuestra Web App. A la par con esta aplicación Web crearemos una versión de escritorio de nuestro sistema usando el plugin de adobe AIR el mismo que tendremos que compilarlo usando el SDK de Flex 3.

El paradigma de programación en el cual nos basaremos será Programación Orientada a Objetos, con lo que podremos crear nuestra aplicación con características como herencia, abstracción, encapsulamiento, modularidad etc., y como medio de comunicación de datos usaremos los Web Services, para realizar el consumo de la información que generemos por medio de nuestro sitio. Implementaremos API´s de Google como son Google Maps y Google Languaje para demostrar la versatilidad con la que se acoplan las diferentes aplicaciones RIA´s entre sí.

Sabiendo que la tecnología va a la par con la necesidad de las personas de tener movilidad y comunicación en todo momento, en la mayoría de los dispositivos que actual mente se usan, lo que queremos es dejar una imagen en la mente de quienes se interesen por este tema, para que vean el futuro que nosotros hemos podido avizorar mediante el estudio de estas tecnologías.

CAPÍTULO I

CAPÍTULO 1

INVESTIGACIÓN TEÓRICA

1.1. Introducción

En este capítulo trataremos sobre las tecnologías de la WEB, su origen, desarrollo y evolución a lo largo de estos años, la importancia que esta tiene hoy en día y el gran impacto que ha causado en la vida cotidiana.

En la actualidad la demanda de aplicaciones basadas en Internet continúa imparable, las exigencias de los usuarios finales y de las empresas obligan a mejorar la difusión y el comportamiento de los sistemas web, obligando a los desarrolladores a crear sitios en menor tiempo y con mejor tecnología.

Las necesidades actuales de información están llevando a muchas empresas a buscar modelos de aplicaciones de internet más ricas (RIA's), modelos que combinan la potencia de los ordenadores actuales, con la gran velocidad de conexión que se ofrece por parte de las operadoras y sitios web con innovadores diseños, gran capacidad de procesamiento de Información y adaptación en los diferentes sistemas operativos y en los diferentes equipos que van desde computadores, teléfonos celulares y los más inusuales gadgets.

1.2. Evolución de las aplicaciones informáticas

En los primeros días de las aplicaciones informáticas para las empresas, todos los procesos tenían lugar en el ordenador central y el cliente no tenía ningún papel excepto visualizar información del servidor y aceptar las entradas de los usuarios. Esto se debía al elevado costo del procesamiento. No era posible tener clientes importantes en distintos lugares, así que todo el procesamiento estaba consolidado y estos "falsos terminales" (terminales sin procesador) proporcionaban la interacción con el usuario.

A medida que la capacidad de procesamiento y de memoria se volvía más barata, los falsos terminales fueron reemplazados por microordenadores (ordenadores personales). Con todos estos avances se podían ejecutar de forma independiente aplicaciones de escritorio como los procesadores de texto y las hojas de cálculo, ya que no era necesario ningún servidor.

Uno de los desafíos con los que se encontraban las empresas con microordenadores era la falta de datos centralizados por lo que había más desafíos a las reglas de las empresas centralizadas en cuanto a la sincronización de datos se refiere.
Para ayudar a resolver esta situación salieron al mercado plataformas que pretendían combinar la fuerza de los microordenadores con las de los ordenadores centrales, dando lugar al nacimiento de los sistemas cliente/servidor, esta plataforma proporcionaba a los usuarios finales la potencia y la facilidad de utilización de los microordenadores, el nuevo desafío que planteaba el sistema cliente/servidor era la distribución.

En el momento en el que era necesario realizar algún cambio en las aplicaciones del cliente, los departamentos de informática tenían que reinstalar manualmente o actualizar el software de cada ordenador de forma individual. Muchas empresas se encontraban con que necesitaban

trabajadores a tiempo completo, cuya responsabilidad principal era el mantenimiento del software en el ordenador del usuario final.

Con el explosivo desarrollo de internet en la década de los 90, las empresas tenían a su disposición un nuevo modelo de aplicaciones. Este modelo funcionaba con un navegador de páginas Web, cuyo trabajo principal era interpretar el lenguaje HTML y volver a enviar peticiones a un servidor de aplicaciones, que escribía las páginas de forma dinámica y las enviaba al cliente.

Este modelo se denomina "**Arquitectura basada en páginas**" y resolvía de una manera correcta el problema de la distribución en los días del cliente/servidor; la aplicación se descargaba desde el servidor cada vez que un usuario final lo necesitaba, de forma que las actualizaciones podían hacerse en un único sitio centralizado y distribuirse de forma automática a toda la red de usuarios.

Este modelo era, y continua siendo, satisfactorio para muchas aplicaciones; sin embargo, también tiene desventajas y limitaciones importantes.

En realidad, las aplicaciones de internet tienen un gran parecido con las aplicaciones de los ordenadores centrales en la medida que todo el procesamiento estaba centralizado en el servidor, y el cliente solo interpretaba los datos y capturaba la alimentación hecha por el usuario.

Los problemas más importantes estaban relacionados con la interfaz de usuario. Muchas de las comodidades que los usuarios finales habían aceptado durante la década anterior se habían perdido, y la interfaz de usuario estaba limitada a las prestaciones del lenguaje HTML.

Por ejemplo, el software de los ordenadores de sobremesa además de usar las aplicaciones cliente/servidor utilizaba de forma frecuente la característica de arrastrar y soltar.

Sin embargo, las aplicaciones HTML casi nunca usan esta característica por ser muy compleja y el no existir soporte compatible con todos los servidores para los elementos DHTML (*Dynamic HTML*, HTML dinámico) necesarios para implementar la característica arrastrar y soltar en una solución pura HTML/DHTML.

En la mayoría de los casos, el grado de sofisticación total de las soluciones que podrían construirse y repartirse estaba enormemente reducido. Aunque la Web ha ofrecido grandes mejoras en las aplicaciones en cuanto a la facilidad y la rapidez de uso, las aplicaciones basadas en la Web para las empresas han dado un paso atrás porque tenían que adaptarse a las limitaciones de la arquitectura Web: HTML y HTTP (Protocolo de transferencia de Hipertexto).

En la actualidad la demanda de aplicaciones basadas en Internet continúa creciendo y por lo general es bastante diferente a la demanda de la década de los noventa. Los usuarios finales y las empresas exigen más de sus inversiones en tecnología Internet.
La capacidad para distribuir valores reales a los usuarios finales está obligando a muchas empresas a buscar modelos de aplicaciones de internet más ricas, modelos que combinan la potencia de los ordenadores de sobremesa tradicionales, ricos en dispositivos, con la utilización y naturaleza rica en contenido de las aplicaciones web.
Cuando las aplicaciones de internet empezaron a utilizarse para mejorar la funcionalidad de la parte central de la empresa, el mantenimiento de esta aplicación se vuelve fundamental.

Este mantenimiento está relacionado con la arquitectura de la aplicación. Lamentablemente, muchas aplicaciones web se han construido con pocas referencias a los principios de la arquitectura de la aplicación, y por tanto es difícil mantenerlas y ampliarlas.

Hoy en día, es más fácil construir una arquitectura sólida para una aplicación con una separación total entre clientes, acceso a los datos y áreas de presentación. Con la introducción de elementos como los servicios Web, el concepto de SOA (*Service-Oriented Architecture*, Arquitectura Orientada A Servicios) se ha vuelto más factible para las aplicaciones basadas en la Web.

Para satisfacer la demanda de las empresas, las RIA deberían ser capaces de:

- Proporcionar un tiempo de ejecución eficiente y de alto rendimiento para ejecutar el código, el contenido y las comunicaciones.

- Proporcionar modelos de objetos potentes y extensibles para facilitar la interactividad. Los navegadores Web han avanzado mucho en los últimos años en su capacidad para admitir la interactividad a través de los DOM (*Document Object Mode*l, Modelo de Objetos de Documento) a través de JavaScript y DHTML, pero que todavía carecen del soporte estandarizado compatible con todos los servidores y que se puedan utilizar en diferentes navegadores. Construir RIA con estas herramientas para que funcionen en muchos navegadores y sistemas operativos implica la construcción de muchas versiones de la misma aplicación.

- Posibilitar la utilización de los objetos *Server-Side* (Del Servidor) mediante el uso de servicios Web u otras tecnologías similares. La promesa de las RIA incluye la posibilidad de separar totalmente la lógica de la presentación y de las interfaces de usuario de la lógica de la aplicación almacenada en el servidor.

- Posibilitar la utilización de la aplicación sin conexión a Internet. Los ordenadores portátiles y otros dispositivos portátiles tienen cada vez más popularidad, y una de las limitaciones de las aplicaciones de Internet es el requisito de que la máquina que ejecuta la aplicación tenga que estar conectada a Internet. Aunque los usuarios puedan estar conectados la mayor parte del tiempo, existen ocasiones en las que una conexión a Internet no es posible. Una RIA eficaz debería permitir a los usuarios ser productivos con o sin conexión a internet.

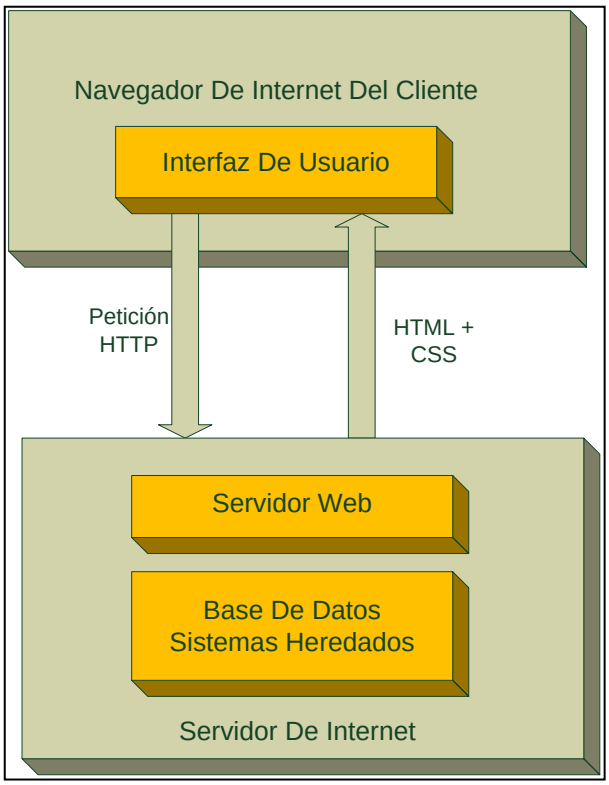
1.3. Aplicaciones WEB una visión del pasado

En los comienzos de la *World Wide Web*, la información estaba contenida en páginas web interconectadas por vínculos estáticos limitada a su lectura, y para acceder a información complementaria hacía falta de un enlace a una dirección URL. Desde entonces, los usuarios pueden mantener cierta interactividad con una página WEB, como es el ingreso de datos a un formulario, procesamiento de datos, etc. Las páginas WEB estaban escritas en lenguaje HTML (HyperText Markup Language), el cual nos permite escribir páginas web estáticas mediante el uso de etiquetas (tags), para desarrollar la estructura de contenido en forma de texto, e incluso complementar el texto con contenidos como imágenes, videos, sonidos, etc.

1.3.1. Aplicaciones clásicas de internet

El siguiente paso que se dio fue el comenzar a utilizar aplicaciones instaladas en el servidor WEB, llamados Interfaz de entrada Común o CGI (*Common Gateway Interface*) por sus siglas en inglés, los cuales permiten a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web .

La forma de relacionarse entre estas páginas es por medio de enlaces, que crean las bases de una estructura del hipertexto.



Fi **Figura 1.1 Modelo de una aplicación web**

La interacción con la interfaz del usuario es interpretada como una petición HTTP (HyperText Transfer Protocol), la misma que es enviada al servidor web, atendiendo de esta forma los requerimientos de las bases de datos y también realizando la actualización de la página, enviando al navegador el código HTTP y CSS (Cascade Style Sheets) necesarios para dicha tarea.

1.4. Dejar atrás la arquitectura basada en página

Para los desarrolladores Web con experiencia uno de los mayores desafíos a la hora de construir una RIA es apartarse de la estructura basada en la página. Las aplicaciones Web tradicionales se centran en el concepto de página Web.

Sin tener en cuenta las tecnologías del Server/side utilizadas (si se utiliza una), el proceso es el siguiente:

1. El usuario abre un navegador y solicita una página a un servidor Web.

2. El servidor Web recibe la solicitud.

3. (Opcional) El servidor Web entrega la petición a un servidor de aplicaciones para ensamblar la página de forma dinámica

4. (Opcional) El servidor Web recupera la página estática de un sistema de archivos.

5. El servidor web envía la página dinámica o estática, devuelta al navegador.

6. El servidor coloca la página en el lugar de la que se estaba visualizando antes. Incluso en aquellas situaciones en las que el contenido de las páginas anteriores es idéntico a la nueva página, es necesario reenviar la nueva página completa al navegador para que este la muestre. Ésta es una de las deficiencias de las aplicaciones Web tradicionales: cada interacción con el usuario requiere la carga de una nueva página en el navegador.

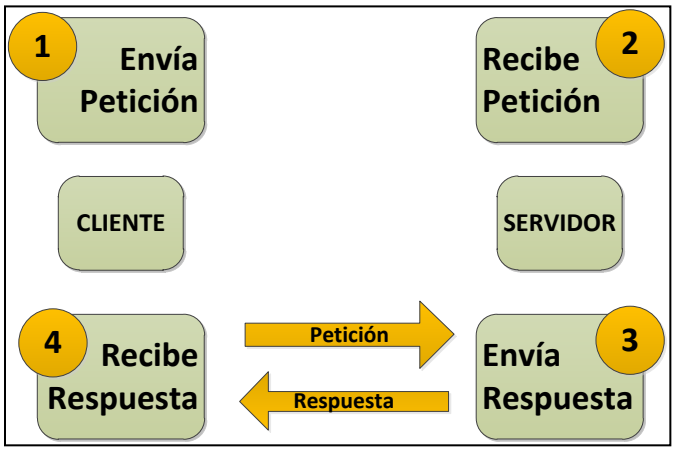


Figura 1. 2 Petición Cliente/Servidor

Uno de los objetivos más importantes de las RIA es reducir la cantidad de datos adicionales transmitidos en cada petición. En lugar de descargar una página entera, ¿Por qué no descargar únicamente los datos que cambian y actualizar la página que está examinando el usuario? Ésta es la forma en la que trabajan los ordenadores de escritorio estándar o las aplicaciones cliente/servidor.

Aunque este objetivo parece sencillo y es aceptado inmediatamente por los desarrolladores que tienen su primer contacto con el desarrollo de las RIA, es frecuente que los programadores Web mantengan un modo de pensar basada en la página cuando se enfrentan a las RIA y se esfuerzan en afrontar desafíos como "Mantener estado" desde un mundo basado en la página. Por ejemplo, luego de que los usuarios accedan a internet, ¿Cómo sabemos quiénes son y qué pueden hacer cuando navegan en la aplicación?

El concepto de mantener estado fue un desafío introducido por las aplicaciones basadas en la Web. HTTP fue diseñado como un protocolo *stateless (sin estado)* en el que cada petición al servidor era una unidad atómica que no sabía nada de las peticiones previas. Esta naturaleza stateless de la Web permitía una gran eficiencia y redundancia porque la conexión entre navegador y servidor no necesitaba estar abierta. Cada

nueva petición de página se mantiene solamente mientras el servidor recupera y envía la página, permitiendo que un único servidor maneje más peticiones de forma simultánea.

La naturaleza *stateless* de la Web añadía desafíos para los desarrolladores de aplicaciones. Por lo general, las aplicaciones tienen que recordar la información del usuario, permiso de acceso, artículos añadidos al carro de la compra, etc. Sin la capacidad para recordar esta información de una petición a otra, el verdadero desarrollo de la aplicación no será posible.
Para ayudar a solucionar este problema se han puesto en práctica una serie de soluciones que giran en torno a un único símbolo que se envía de vuelta al servidor en cada petición (a menudo en forma de *cookies,* que son pequeños archivos de texto que contienen pequeños identificadores específicos de la aplicación para un usuario individual) y que hacen que el servidor almacene la información del usuario.

A diferencia de las aplicaciones Web tradicionales, las RIA pueden evitar muchos de estos problemas. La aplicación permanece en la memoria RAM del cliente mientras es utilizada, por lo que en lugar de ser cargada y descargada como un modelo basado en la página, se pueden establecer variables una vez y acceder a ellas durante el ciclo de vida de la aplicación.

Una propuesta para mejorar el estado es considerarlo como uno de los muchos lugares en los que la construcción de las aplicaciones requiere un modo de pensar ligeramente diferente al del desarrollo de las aplicaciones Web. En realidad, las RIA basadas en la Web tienen más similitudes con las aplicaciones cliente/servidor que con las aplicaciones Web.

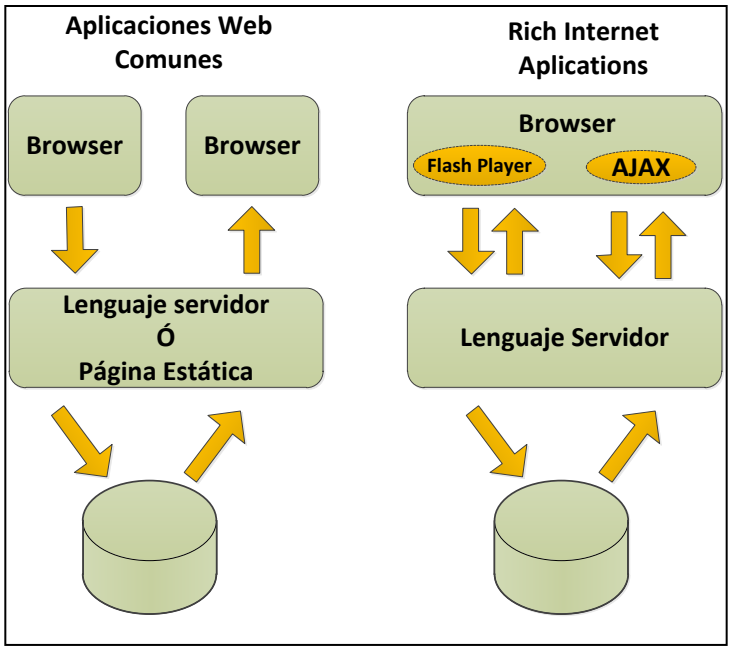


Figura 1. 3 Aplicaciones Web vs RIA

1.5 Comercio electrónico "Un nuevo enfoque de ventas"

El comercio electrónico es el nuevo marco de negocios en el que se desarrollan cada vez más las operaciones mercantiles. En el comercio electrónico convergen tecnologías y aplicaciones que ya existían aisladamente y otras como el intercambio electrónico de datos, la publicidad en Internet, las Intranet o las compras electrónicas, pero que tienen en común el uso de las telecomunicaciones.

Internet ha abierto un gran número de nuevas posibilidades para las empresas con iniciativa de crecimiento. Una de las más importantes es la posibilidad de habilitar un nuevo canal de ventas para promover y vender productos y servicios, o dicho de otra manera, hacer comercio electrónico.

Para hacer comercio electrónico es necesario contar con una plataforma tecnológica integral y confiable que incorpore catálogos electrónicos, carrito de compras, sistemas electrónicos de pago seguros, sistemas de control de órdenes, acceso dinámico a inventarios, sistemas de soporte a clientes, etc.

Se debe determinar cuáles son los beneficios que la empresa desea obtener al habilitar un canal de ventas en la Red y deberán tenerse en cuenta el gran número de retos que se estarán adquiriendo.

1.5.1. Beneficios

1. Acceso a un mercado mundial

Una de las principales características de Internet es su alcance global y el modo de introducirse en prácticamente todos los países del mundo. Una buena estrategia de posicionamiento en un sitio Web en Internet ayudará a la empresa a incrementar el tamaño de su mercado de forma exponencial. Además de la posibilidad de comercializar sus productos en otros países utilizando la misma plataforma (sitio Web), se puede además tener una comunicación más cercana y efectiva con clientes, proveedores y socios tanto nacionales como extranjeros.

2. Negocios las 24 horas del día, todo el año

Un sitio Web le da la posibilidad a tu empresa de mantener una oficina virtual o departamento de ventas que trabaja los 365 días del año las 24 horas del día con un costo relativamente bajo, comparado con los gastos que conlleva mantener una oficina tradicional con gastos de empleados, equipo, servicios, etc. Tener esta gran disponibilidad para hacer negocios se vuelve particularmente importante para empresas con mercados

internacionales. Debido a los diferentes usos horarios, muchos de los clientes de otras latitudes pueden hacer negocios con nuestra empresa incluso mientras los empleados duermen. Además, existe un buen porcentaje de usuarios de Internet que les gusta navegar hasta altas horas de la noche y que aprovecha parte de este tiempo para adquirir productos y servicios.

3. Comodidad para los clientes

Muchas personas encuentran mucho más cómodo adquirir sus productos y servicios desde su propio domicilio a través del teléfono o utilizando Internet. Las razones son muchas, pero entre las más importantes se encuentran: el tiempo, el costo y el esfuerzo. En cuanto al tiempo se debe considerar cuanto ocupa una persona buscando un producto en almacenes; en cuanto al esfuerzo, la necesidad de ir físicamente al lugar y posponer otras actividades posiblemente igual de importantes que hacer la compra. Conforme avanzan los años se va incrementando aún más la cantidad de actividades que una persona debe llevar a cabo, de ahí la importancia de proveer a nuestros clientes de herramientas eficientes que proporciones ahorros significativos en cuanto a los factores que se mencionaron.

4. Ventaja competitiva

Una empresa que comienza antes que sus competidores a utilizar el Comercio Electrónico de manera estratégica, está obteniendo consecuentemente una gran ventaja competitiva ya que está utilizando un canal extra para comercializar sus productos y servicios, además la parte proporcional de su segmento de mercado en Internet será mucho mayor que en el mercado tradicional donde ya se encuentran todos sus competidores.

1.6. RIA y Comercio Electrónico

Gracias a las nuevas tecnologías de internet se ha podido enriquecer el comercio electrónico hoy en día, haciendo mucho más atractivas y fáciles las aplicaciones, dirigiéndose directamente a las necesidades clave que tienen las compañías ayudando a los clientes a ver y entender lo que están comprando.

Un desarrollador de RIA integra animaciones 2D/3D, gráficos ricos, encuestas interactivas, vídeo y clips de audio para aumentar la interacción entre las aplicaciones web y los usuarios finales. Ofrecen soluciones completas de RIA para carritos de la compra en línea y sitios web de comercio electrónico, donde los usuarios pueden navegar fácilmente a través de la tienda en línea en el clic de un botón para seleccionar y comprar un producto de la elección.

La grandeza de RIA es que dichos datos se transfieren rápidamente de una forma a otra y con el servidor, sin tener el usuario final espera. Esto reduce la congestión, acelera la interactividad y aumenta el nivel de rendimiento de la aplicación web. Los desarrolladores de Flex o Silverlight crean soluciones RIA efectivas para la creación de redes en línea, el comercio, los sitios de compras, que requieren alto nivel de interacción del cliente final. Estas tecnologías permiten a los desarrolladores crear aplicaciones enriquecidas, robustas y escalables que pueden manejar fácilmente el flujo de tráfico alto con facilidad, sin colapsar debido al acceso simultáneo de la aplicación web, a partir de un gran número de usuarios.

Rich Internet Application (RIA) ha creado una nueva generación de aplicaciones web que ofrecen los datos de negocio a través de Internet para que las organizaciones empresariales puedan interactuar sin problemas con sus clientes en todo el mundo para aumentar sus ingresos de negocios.

En comparación con el alto precio del software tradicional de inteligencia de negocios, RIA ofrece una opción más barata pero de gran alcance para las organizaciones empresariales. Les da la oportunidad de explorar las vías de nuevos negocios, que no pudieron mantenerse y cerraron por falta de tecnología accesible. La disponibilidad de soluciones personalizadas RIA han abierto la puerta a millones de nuevos usuarios, que en el pasado era difícil hacerlo debido a la falta de recursos adecuados, la tecnología, y los altos costos.

Los avances en el desarrollo de la tecnología RIA han dado poder a los proveedores de servicios para ofrecer soluciones de internet enriquecidas más innovadoras a las organizaciones empresariales. Esto ha ayudado a elevar su nivel de rendimiento y mantener el ritmo de las crecientes expectativas de sus clientes. Como resultado, son capaces de retener a sus clientes mayores y añadir otros nuevos, incluso en medio de una dura competencia.

1.7. Conclusión

En el presente capítulo se pudo mostrar de forma general lo que era en el pasado la Web y actualmente lo que significa trabajar en Internet, con todos los requerimientos y necesidades que se han ido creando por todos los usuarios que día a día hacen interminable la expansión de la web como la conocemos, es por esta razón que vimos imprescindible explicar de forma general tanto su comportamiento, arquitectura y funcionamiento para poder orientarnos en los siguientes capítulos a cual está siendo su evolución en cuanto a estos mismos tópicos demostrando coherencia entre estos conceptos.

CAPÍTULO II

CAPÍTULO 2

APLICACIONES DE INTERNET ENRIQUECIDAS (RIA)

2.1 Introducción

En el siguiente capítulo profundizaremos el concepto de RIA´s siendo que estas son aplicaciones web que tienen tanto características como funcionalidades de aplicaciones de escritorio, con la diferencia de que estas no necesitan instalar ningún programa en el ordenador del usuario para funcionar (salvo sea caso particular), sino, que son accesibles desde un navegador web, dando la característica de crossPlataform a las RIA´s, es decir, que no importa la plataforma que el usuario utilice, estas aplicaciones siempre se verán y funcionaran de igual manera.

Además, estudiaremos ejemplos, ventajas de las RIA´s así como las herramientas que se usan para crearlas y la categorización que se le da a estas de acuerdo a la magnitud de la actividad delegada, ya sea del lado del servidor como del lado del cliente.

2.2 Aplicaciones De Internet Enriquecidas (RIA) "El futuro de la Web empieza"

RIA, acrónimo de ***R**ich **I**nternet **A**pplications* (Aplicaciones de Internet Enriquecidas) son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web.

Las aplicaciones dinámicas de Internet, (RIA) son el siguiente escalón en los requerimientos de los usuarios de la Web para describir la nueva generación de recursos tecnológicos disponibles en Internet y su impacto en la sociedad.

Por esto describiremos sus características principales y tecnologías relacionadas, tomando en cuenta algunos ejemplos populares de la Web.

- En los entornos RIA no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos.

- Las capacidades multimedia son totales gracias a que estos entornos tienen reproductores internos y no hace falta ningún reproductor del Sistema Operativo del usuario.

- Hay muchas herramientas para la creación de entornos RIA. Entre estas se puede mencionar las plataformas Adobe Flash, Adobe Flex y Adobe
  AIR de Adobe, uniPaaS de MagicSoftware, AJAX, OpenLaszlo, Silverlig htMicrosoft, JavaFX Script de Sun Microsystems, Bindows de MB Technologies y JavaScript.

- En un comienzo, el modelo utilizado por la mayoría de las aplicaciones de Internet era muy similar al de las instalaciones centralizadas. En las cuales un computador central (Mainframe) era el responsable de todo el procesamiento y un conjunto de terminales conectadas al mismo, permitían a los usuarios interactuar con las aplicaciones.

- La principal desventaja de este modelo es que toda la interacción con la aplicación, es procesada íntegramente en el servidor. Esto significa que, cada interacción del usuario con la aplicación se traduce en una petición al servidor, la cual es procesada por el mismo y posteriormente, la respuesta es enviada nuevamente al cliente, lo que generalmente produce la actualización de toda la página web, por parte del navegador de Internet.

Una de las principales características de las (RIA), es la delegación de una parte del procesamiento en el cliente, sobre todo, lo referente a la interacción directa con el usuario.

Esta distribución de las responsabilidades entre cliente y servidor presenta las siguientes ventajas:

- **Mayor interactividad**

  Al delegarse parte del procesamiento en el cliente, el usuario suele percibir una respuesta más inmediata al interactuar con la aplicación, que la obtenida con las aplicaciones web tradicionales, que realizan todo el procesamiento en el servidor.

- **Actualización permanente**

  Parte del código de la aplicación se transfiere directamente al navegador al arrancar la misma y suele actualizarse frecuentemente a medida que la misma se está ejecutando, por esta razón no es necesaria ninguna actualización de versiones.

- **Ejecución sin instalación previa**

  El código de la aplicación se va descargando de Internet mientras la misma se ejecuta, por lo tanto, no es necesaria una instalación previa de la aplicación en la computadora del usuario.

- **Distribución y ejecución multiplataforma**

  Lo único que necesita el usuario para utilizarlas es un navegador de Internet. El código de las RIA es interpretado por el propio navegador, lo que lo hace independiente del sistema operativo que utilice el usuario. Por lo tanto, no es necesario contar con diferentes versiones para cada sistema operativo. Por supuesto que estas ventajas tienen un costo asociado.

Para que las aplicaciones dinámicas de Internet no necesiten instalación previa, el código de las mismas debe transferirse en algún momento desde el servidor hasta el navegador del usuario. Y si bien es posible utilizar técnicas de compresión, de transferencia incremental y de almacenamiento intermedio para minimizar el tiempo de transferencia, puede existir cierta latencia que dependerá diversos factores, como la

velocidad de la conexión y la disponibilidad de atención que presente el servidor.

Por otra parte, para independizar a las aplicaciones del sistema operativo del usuario, se delega en el navegador la ejecución de código interpretado, usualmente JavaScript 2. La utilización de código interpretado brinda una gran flexibilidad, permitiendo incluso su modificación en tiempo de ejecución, como veremos más adelante. Si bien estos lenguajes interpretados pueden presentar una velocidad de procesamiento inferior a la de los lenguajes compilados, esto es cada vez menos perceptible por el usuario, ya que las velocidades de comunicación y procesamiento son grandes y continúan en franco crecimiento.

2.3 Ejemplos De Aplicaciones Air

Entre algunos de los ejemplos más populares en la Web tenemos:

Ebay Desktop: El eBay Desktop, la alternativa en el escritorio y en el ordenador para usar eBay fuera de su website, y sin tener que estar pendiente de una conexión a Internet.

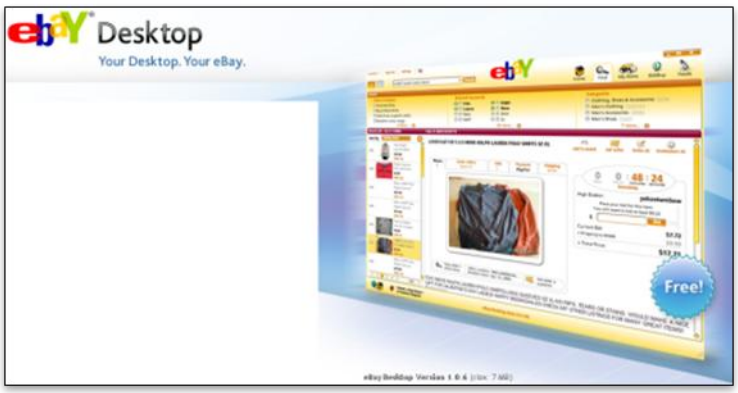


Figura 2.1 Ebay Desktop

NASDAQ: La bolsa de valores electrónica automatizada más grande de Estados Unidos. Con más de 3,800 compañías y corporaciones, tiene más volumen de intercambio por hora que cualquier otra bolsa de valores en el mundo.

Figura 2. 2 NASDAQ

Google Earth: Permite visualizar cualquier parte de la Tierra con imágenes de satélites, mapas, relieves, edificios en 3D desde galaxias del espacio exterior hasta cañones en los océanos. Se puede explorar un rico contenido geográfico, guardar los lugares que se visita y compartirlos con otras personas.

Figura 2. 3 Http://earth.google.es/

Nick.com: Es un sitio web operado por Nickelodeon. El sitio está realizado en gran parte por Adobe Flash, incluye centenares de características y funciones que solo las Aplicaciones de Internet Dinámicas nos pueden ofrecer.

Figura 2. 4 Http://www.nick.com/

Air Force (Fuerza Aérea De Estados Unido)

Figura 2. 5 Http://www.airforce.com/

Estos ejemplos nos pueden ayudar a aclarar cuál es el aporte de las RIA en el desarrollo de nuevos sitios de Internet.

Las aplicaciones dinámicas de Internet comparten una gran cantidad de características con las aplicaciones de escritorio, en cuanto a recursos disponibles y nivel de interactividad.

Para dar una idea de la diferencia, consideremos la comparación entre un sitio web que permite acceder a un conjunto de figuras con mapas estáticos, con una aplicación como Google Maps que le permite a los usuarios interactuar directamente con el mapa, desplazándolo, agrandando una zona o buscando una localización en particular, en tiempo real.

Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegado muchas veces, a recargar la misma página con un mínimo cambio.

Otra de las desventajas de las tradicionales aplicaciones Web es la poca capacidad multimedia que posee. Para ver un vídeo es necesario usar un programa externo para su reproducción.

En la actualidad, los desarrolladores tienen muchas más opciones tecnológicas para construir una RIA. Las opciones más populares son las opciones basadas en HTML como AJAX (*Asynchronous JavaScript and XML*), y las opciones basadas en plugin como Adobe Flash, Adobe Flex y otros que se ejecutan en Flash Player. También han llegado nuevas opciones de Microsoft, como Windows Presentation Fundation (WPF), Silverlight y XAML (Extensible Application Markup Languajes) que ya son una realidad.

Existen cuatro formas diferentes de ejecución en las que se basan las aplicaciones RIA actuales: AJAX, Flash Player, Windows Presentation Fundation, y Java, utilizado por AWT (Abstract Window Toolkit), Swing y Eclipse RCP. Parece que las soluciones de Java y de WPF están más enfocadas a las aplicaciones de los ordenadores de sobremesa que las RIA, aunque también podrían utilizarse con las RIA.

2.4 Ventajas de las Aplicaciones Ricas de Internet

A finales de la década de los noventas se produjo el auge de las páginas punto com, sin embargo hoy en día las empresas ya no invierten en tecnologías de Internet sólo porque esté de moda. Para tener éxito, una nueva tecnología necesita demostrar un retorno de inversión real y un valor añadido verdadero. Las RIA lo consiguen en varios niveles reducen, reducen los costes de desarrollo y añaden valor a la empresa.

2.4.1 Empresas

Es más fácil hacer trabajos con esta tecnología para los usuarios, por lo que el número de transacciones realizadas aumenta. Este aumento se produce en muchas industrias y puede ser cuantificado con números por los clientes, como el aumento de productividad debido a la utilización de aplicaciones de intranet o el aumento del porcentaje de compradores online.

2.4.2 Organismos de las Tecnologías de la Información

Alejarse de la arquitectura basada en la página reduce la carga de los servidores Web y reduce el tráfico total de la Red. En lugar de transmitir páginas enteras una y otra vez, la aplicación completa se descarga una sola vez y la única comunicación desde y hacia el servidor es la de los datos que aparecen en la página.

Si se reduce la carga del servidor y el tráfico de la red, los costes de infraestructura pueden ser notablemente inferiores. Las RIA que se han desarrollado utilizando los principios de la arquitectura sólida y las prácticas más adecuadas también pueden aumentar el mantenimiento de una aplicación, además de reducir enormemente el tiempo de desarrollo necesario para construirla.

2.4.3 Usuarios finales

Los usuarios finales son los más beneficiados de las ventajas de las RIA. Una RIA bien diseñada reduce el nivel de frustración del usuario porque ya no es necesario navegar por varias páginas para encontrar lo que necesita o esperar a que una nueva página se cargue antes de continuar con la actividad. Además el tiempo que el usuario requiere para aprender el funcionamiento de la página se ve muy reducido por lo que este se encuentra más capacitado.

Hoy en día existen un gran número de aplicaciones que no serían posibles sin el concepto de las RIA, como las aplicaciones de Harley Davidson Motorcycle Configurator y las aplicaciones Virtual Ubiquity de Buzzword, Google Earth, Aol Music etc.

2.5 Tecnologías Aplicadas a la creación de RIA

2.5.1 Tecnologías Asincrónicas

El término AJAX (Asynchronous JavaScript And XML), fue introducido por primera vez por Jesse James Garrett en Febrero de 2005, para denominar a un conjunto de técnicas inter-relacionadas, tales como XHTML, CSS, DOM, XML, HTTP y JavaScript, comúnmente utilizadas para crear aplicaciones web interactivas.

La utilización de AJAX para la creación de aplicaciones web, permite lograr un mayor grado de interactividad, ya que disminuye la necesidad de refrescar toda la información contenida en la página Web. Esto reduce notablemente la cantidad de información transmitida desde y hacia el servidor, con el consiguiente incremento de velocidad en las respuestas.

En la figura 2.6 se puede apreciar el modelo que representa la interacción asincrónica del cliente con el servidor, por medio de AJAX. A diferencia del modelo clásico, vemos que aparece una nueva capa, representada por el motor AJAX, que permite realizar una parte del procesamiento directamente del lado del cliente.

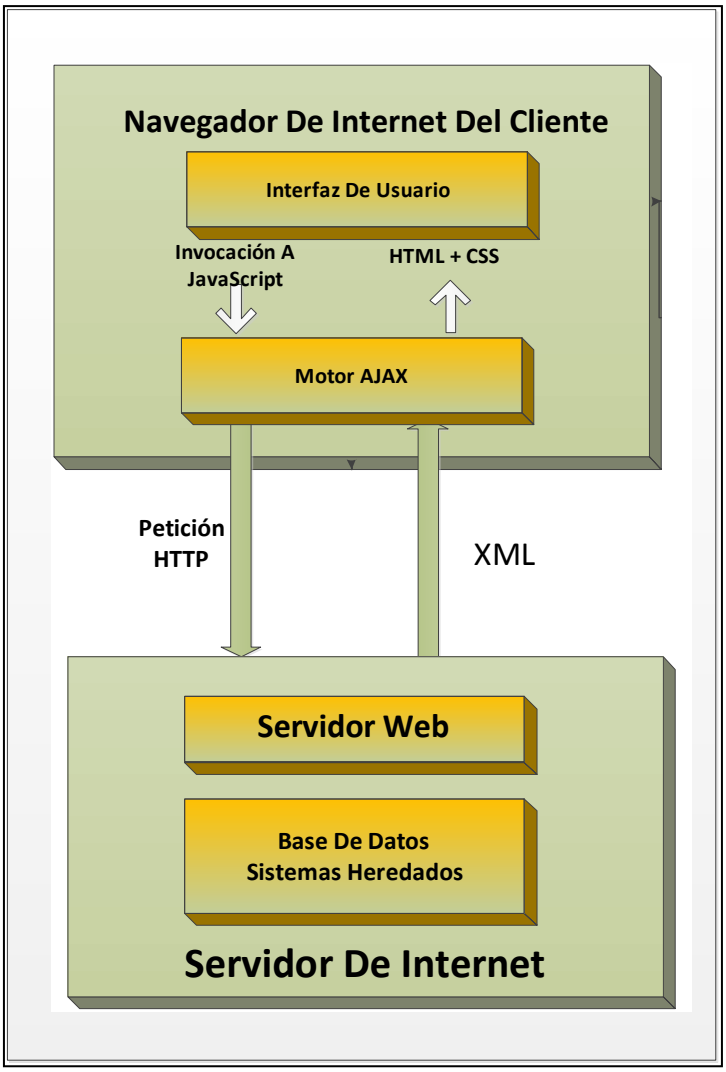


Figura 2. 6 Modelo de una aplicación Web AJAX

Por lo general, este procesamiento suele estar relacionado con la interfaz de usuario. Es decir, todas aquellas peticiones que no requieren

del acceso a datos almacenados en el servidor, son resueltas directamente por el motor AJAX.

Un mecanismo de interacción similar fue propuesto por Netscape en 2003. En un documento llamado "Inner Browsing", se plantea la posibilidad de realizar toda la navegación del sitio en una sola página Web. Esta página se actualizaría por partes, según las necesidades del usuario, quebrando así, con el paradigma de navegación página por página de los sitios web tradicionales.
Este cambio creó un nuevo paradigma, **"de las aplicaciones dinámicas de Internet"**.

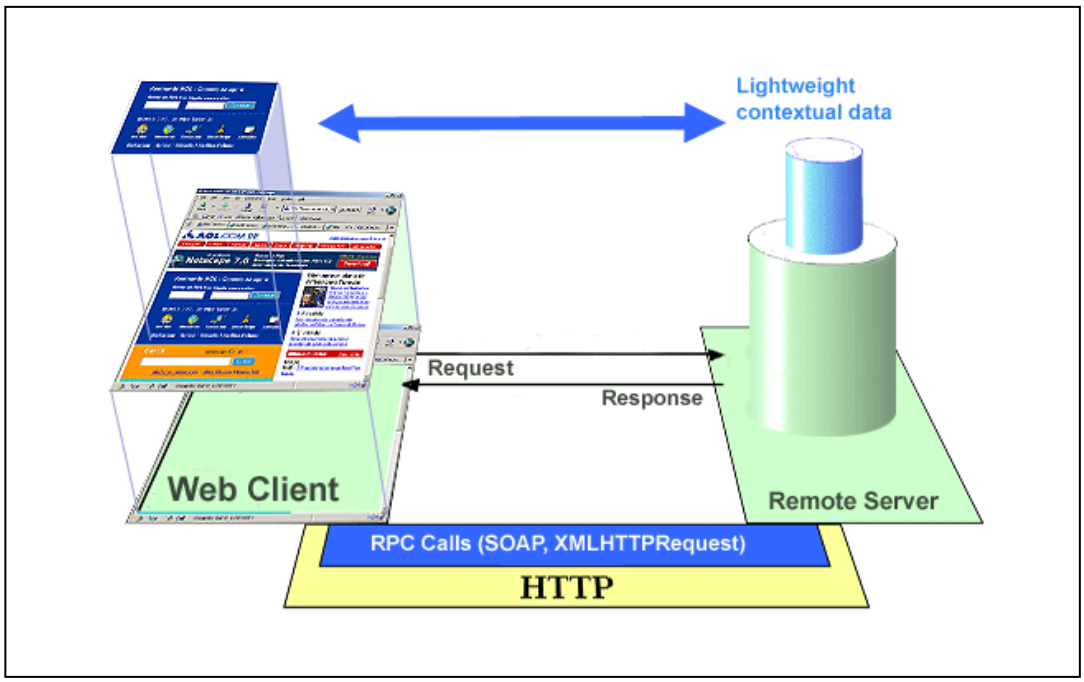


Figura 2. 7 Modelo de nuevo paradigma de las aplicaciones dinámicas de Internet

Imagen de: *postgrado.info.unlp.edu.ar/Carrera/Especializaciones/.../LIZARRALDE.pdf*

En este tipo de aplicaciones, la interfaz de usuario, que puede presentar el mismo aspecto que una aplicación de escritorio, no se actualiza por completo, sino sólo en aquellas partes que han modificado su contenido.

Actualmente, este conjunto de tecnologías agrupadas bajo el nombre de AJAX, permite que las aplicaciones web posean recursos, que hasta hace poco tiempo, sólo estaban disponibles en las aplicaciones de escritorio.
A continuación se presentan las principales características y lenguajes presentes en la construcción de RIA.

2.5.2 AJAX: Asynchronous JavaScript and XML

Una de las opciones más fáciles de entender es AJAX, que es el acrónimo de JavaScript y XML Asíncronos. AJAX se basa en herramientas que ya son familiares a los desarrolladores de Web: HTML, DHTML y JavaScript. La idea fundamental en la que se basa AJAX es utilizar JavaScript para actualizar la página sin tener que volver a cargarla.

Un programa JavaScript que se ejecuta en un navegador puede introducir nuevos datos en la página o cambiar su estructura manipulando HTML DOM (*HTML Document Object Model*), sin tener que volver a cargar una nueva página.

La actualización puede llevar consigo la descarga de nuevos datos del Servidor en segundo plano (utilizando XML u otros formatos) o en respuesta a una interacción con el usuario, como hacer clic o mover el cursor.
Al principio las aplicaciones Web utilizaban los *applets* (pequeñas aplicaciones) de Java que se utilizaban para comunicación remota. A medida que se desarrollaron las tecnologías de los navegadores, otros medios como los IFrames (*Inline Frame, marco incorporado*) remplazaron a los *applets.*

Más recientemente, XMLHttpRequest se ha añadido a JavaScript, proporcionando una forma para facilitar la transferencia de datos sin necesidad de una nueva petición de página, applet o IFrame.
Además de la ventaja de que AJAX utilicen elementos que ya son familiares a muchos desarrolladores de aplicaciones Web, AJAX no utiliza plugins externos para ejecutarse. Funciona únicamente con la capacidad del navegador para utilizar JavaScript y DHTML. Sin embargo, la dependencia de JavaScript evidencia una de las nuevas desventajas de AJAX: no funciona si el usuario no tiene JavaScript habilitado en el navegador.

Otra desventaja de AJAX es que tiene distintos niveles de admisión de DHTML y JavaScript dependiendo de los navegadores y de las plataformas.
Para aquellas aplicaciones en las que los usuarios puedan controlarse (como en las aplicaciones de intranet), AJAX puede programarse para admitir un único navegador en una plataforma determinada (muchas personas tienen en la actualidad navegadores y sistemas operativos estandarizados).

Sin embargo cuando las aplicaciones se abren a usuarios más extensos (como extranet y las aplicaciones de internet), las aplicaciones AJAX tienen que probarse y normalmente modificarse, para asegurar que se ejecutan de forma idéntica en todos los navegadores y sistemas operativos.
No es muy probable que AJAX desaparezca a corto plazo, ya que cada día se lanzan al mercado más y más aplicaciones AJAX de alto nivel con enorme éxito, como por ejemplo GOOGLE MAPS.

Hay que tener en cuenta que AJAX no es en la actualidad un modelo de programación en sí mismo. En realidad se trata de un conjunto de

bibliotecas JavaScript. Algunos de estas bibliotecas incluyen componentes reutilizables diseñadas para hacer las tareas más fáciles.

2.5.3 Adobe Flash

Uno de los tiempos de ejecución más competitiva en el espacio de las RIA es la plataforma Adobe Flash. La plataforma Flash es actualmente el competidor más importante de AJAX en las RIA. En su origen fue programado con un plugin para ejecutar animaciones, aunque Flash Player ha evolucionado durante estos años y cada versión añade nuevas características a la vez que todavía se mantienen espacios de utilización muy pequeños. Durante la última década Flash Player se ha convertido casi en omnipresente, con una versión instalada en más del 97 por 100 de todos los navegadores Web.

Desde 2002, Macromedia, en la actualidad parte de Adobe, empezó a centrarse en flash como algo más que una herramienta de animación. Macromedia se encontró que con la omnipresencia de Flash en todos los navegadores y la potencia de su lenguaje de texto (ActionScript), los desarrolladores podían construir aplicaciones basadas totalmente en los navegadores y superar las limitaciones de HTML.

Al centrarse en Flash Player los desarrolladores podían eliminar las incompatibilidades entre plataforma y navegador.
Una de las mejores características de Flash Player es que el contenido y las aplicaciones desarrolladas en cualquier versión de Flash Player se ejecutarán (normalmente) en cualquier plataforma/navegador que admita la versión de Flash Player.

Históricamente, la mayor desventaja en la construcción de aplicaciones para Flash Player era el entorno de creación, que se construía claramente con una herramienta de animación para usuarios que elaboraban contenidos interactivos. Muchos desarrolladores que querían construir RIA para Flash Player fracasaron por la poca familiaridad con las herramientas.

Esto, unido a la falta de materiales disponibles en 2002 para aprender a utilizar Flash como una plataforma de aplicación, hizo que muchos desarrolladores se mantuvieran al margen de la creación de aplicaciones Flash.

Aunque Flash Player es una excelente plataforma para las RIA, la introducción de soluciones como Flex ha simplificado enormemente el proceso de desarrollo; ha reducido el número de RIA desarrolladas directamente en Flash Studio.

2.5.4 Adobe Flex

Flex es un marco de trabajo de código abierto gratuito y altamente productivo para la creación de aplicaciones web expresivas que se implantan coherentemente en exploradores, computadores de mesa y sistemas operativos, aprovechando los tiempos de ejecución de Adobe Flash Player y Adobe AIR. Aunque pueden crearse aplicaciones Flex únicamente mediante el marco de trabajo de Flex, este puede acelerar el desarrollo mediante funciones como códigos inteligentes, depuración interactiva estratificada y diseño visual del aspecto de la interfaz de usuario.

Atendiendo a la necesidad de herramientas más fáciles de usar para construir RIA, Adobe desarrolló un lenguaje y un compilador que permitía a los desarrolladores trabajar con lenguajes familiares con los que su compilador podría crear aplicaciones para ejecutarse en flash

player. En 2004, Macromedia presentó Flex 1.0 (Seguido por Flex 1.5 en 2005).

Adobe continúa este ciclo presentando Flex 2.0, Flex 3.0, Adobe Flash Builder (anteriormente Flex Builder 4), Adobe Flash Burrito (Para aplicaciones celulares), en 2006, 2008, 2009 y 2010 respectivamente. Desde el punto de vista de la arquitectura, las aplicaciones Flex son parecidas a las aplicaciones AJAX en que ambos son capaces de actualizar de forma dinámica la interfaz de usuario y pueden enviar y recibir datos en segundo plano.

Flex ofrece hoy día la siguiente generación de herramientas y servicios que permiten a los desarrolladores construir y usar las RIA en la plataforma Flash. Flex está formado por varias partes:

- **ActionScript 3.0:** Un lenguaje de programación orientado hacia el objeto muy potente, que avanza en las posibilidades de la plataforma Flash. ActionScript 3.0 está diseñada para crear un lenguaje que en teoría es adecuado para construir las RIA rápidamente. Aunque versiones anteriores de ActionScript ofrecían la potencia y la flexibilidad requerida para crear experiencias online atractivas, ActionScript 3.0 avanza aún más en el lenguaje, mejorando la representación y la facilidad de desarrollo para facilitar incluso las aplicaciones más complejas con extensos conjuntos de datos y códigos reutilizables completamente orientados hacia el objeto.

- **Flash Player 10(FP10):** Construido sobre Flash Player 9, esta generación de Flash Player se centra en la mejora de la ejecución del texto. Para favorecer esta mejora FP10 incluye una versión de *AVM* **(ActionScript Virtual Machine**, Máquina

Virtual de ActionScript), nueva y totalmente optimizada, conocida como *AVM2*. AVM2 se construye desde lo más sencillo hasta llegar a trabajar con ActionScript 3.0, la siguiente generación de lenguajes que alimenta Flash Player. La nueva máquina virtual es mucho más rápida y admite informe de errores en el tiempo de ejecución y depuraciones mucho más mejoradas. Flash Player 10 también contiene AVM1, que ejecuta el código ActionScript 1.0, 2.0 y 3.0 para ser compatible con versiones anteriores de contenidos ya existentes. A diferencia de las aplicaciones que se construyen utilizando JavaScript, Flash player es capaz de utilizar un proceso de de compilación JIT (*Just in Time*, Justo a tiempo) que hace que se ejecute en forma más rápida y consuma menos memoria.

- **FLEX SDK:** Utiliza las bases proporcionadas por FP10 y ActionScript 3.0, y la estructura de una biblioteca de clases extensiva que permite a los desarrolladores utilizar fácilmente las prácticas más adecuadas para construir las RIA con éxito.
  Flex utiliza un lenguaje basado en XML denominado MXML que permite a los desarrolladores una forma declarativa para manejar los elementos de una aplicación. Los desarrolladores pueden tener acceso a la estructura a través de Flex Builder o la versión gratuita de Flex SDK, que incluye un compilador de líneas de comandos y un programa de depuración, que hace que los desarrolladores puedan utilizar el editor que refieran y seguir teniendo acceso al compilador o al programa de depuración directamente. En 2007 Adobe anunció el plan de acción de Flex SDK hacia el código abierto, que fue terminado en 2008.

- **Flex Builder 3:** Adobe Flex Builder 3™ es una herramienta de desarrollo basada en Eclipse™ muy productiva que incorpora las siguientes funciones: códigos inteligentes, depuración interactiva estratificada, además del diseño visual del aspecto y comportamiento de la interfaz de usuario de las aplicaciones de Internet sofisticadas (RIA).

  El framework de Flex se utiliza para crear archivos SWF que se reproducirán luego con Adobe Flash Player. Flex fue creado para el uso de los desarrolladores y sigue paradigmas de desarrollo de aplicaciones tradicionales, a diferencia del sistema basado en la línea de tiempo que se usa en Flash.

  Construido sobre los estándares más elevados de la industria, el proyecto de código abierto Eclipse, proporciona un excelente medio para el código y la depuración, es una herramienta de diseño útil y promueve las prácticas más adecuadas en el desarrollo del código y de las aplicaciones. Otra ventaja de la plataforma Eclipse es que proporciona un amplio conjunto rico de posibilidades que se pueden ampliar, por lo que pueden realizarse personalizaciones para ampliar el IDE y adecuarse a las necesidades específicas o a las preferencias de desarrollador.

2.5.5 Windows Presentation Foundation, XAML, Silverlight y Expression

Microsoft ha lanzado al mercado un conjunto de herramientas para ayudar a los desarrolladores en la construcción de las RIA en la plataforma Windows:

- **WPF:** *Windows Presentation Foundation* (Antes conocido como Avalón). Es análogo en estructura a Flash Player y a Flex.

- **XAML:** Extensible Application Markup Language. El lenguaje basado en XML en el que se puede construir aplicaciones WPF. XAML es análogo al lenguaje MXML de Flex.

- **Silverlight:** Un subconjunto basado en la web de WPF, que permite el uso de XAML y JavaScript para crear aplicaciones Web ricas.

- **Microsoft Expression:** Herramienta de diseño profesional para trabajar con XAML y que proporciona interactividad a los diseñadores para crear la interfaz de usuario y el comportamiento visual para las aplicaciones WPF. Es en su mayoría análogo a Flash Studio, ya que es una herramienta diseñada para las aplicaciones WPF.

Con estas herramientas Microsoft, está favoreciendo un flujo de trabajo en el que los diseñadores crean interfaces de usuarios convenientes con Expression (utilizando WPF o Silverlight) y los desarrolladores pueden mejorar las empresas y el acceso lógico a los datos mediante Visual Studio.

Aunque Microsoft ha informado públicamente que admitirá otras plataformas (en concreto con Silverlight), no se ha proporcionado información específica, como qué navegadores y plataformas se admitirán. Es alentador ver que Microsoft finalmente promete proporcionar herramientas para otras aplicaciones que no sean Windows, pero es muy pronto para ver como cumplirán esta promesa.

Microsoft también tiene una herramienta de diseño distinta específica para los diseñadores llamada Expresión. También hay que tener en cuenta que seguramente pase mucho tiempo antes que WPF alcance algún tipo de omnipresencia, ni siquiera en Windows, debido a la gran

descarga necesaria. WPF está disponible para Windows XP, pero necesita una instalación externa de .NET framework 3, y está disponible de forma original en Windows Vista.

2.6 Categorización de las RIA

Las aplicaciones de Internet enriquecidas (RIA) cubren un amplio espectro, y por lo general se diferencian por la magnitud de la actividad delegada, ya sea del lado del servidor como del lado del cliente. Es decir, se podría considerar que, en un extremo se hallan las aplicaciones exclusivamente de escritorio, en las cuales todo el procesamiento se realiza en forma local y, en el otro extremo, se encuentran las aplicaciones web tradicionales, donde todo el procesamiento se realiza en el servidor del sitio web. Si ubicamos en el eje horizontal a los diferentes tipos de aplicaciones, desde las aplicaciones de escritorio hasta las aplicaciones web tradicionales y en el eje vertical representamos el nivel de procesamiento, ya sea del lado del cliente como del servidor, vemos claramente cuál es el lugar que ocupan las aplicaciones dinámicas de Internet.

En la figura 2.8, se observa un área sombreada, cuyo extremo superior izquierdo está representando las aplicaciones con gran nivel de procesamiento del lado del cliente (Cliente Pesado) y en el extremo inferior derecho se hallan las aplicaciones con gran nivel de procesamiento del lado del servidor (Cliente Delgado).

Estos puntos permiten determinar un área rectangular en la que se encuentran enmarcados los diferentes tipos de aplicaciones dinámicas de Internet (RIA).
Sin embargo, a pesar de las diferencias de nivel de procesamiento, ya sea en el cliente o en el servidor, la mayoría de las aplicaciones dinámicas de Internet comparte una característica principal, la complejidad del modelo subyacente es superior al de las aplicaciones tradicionales de Internet.

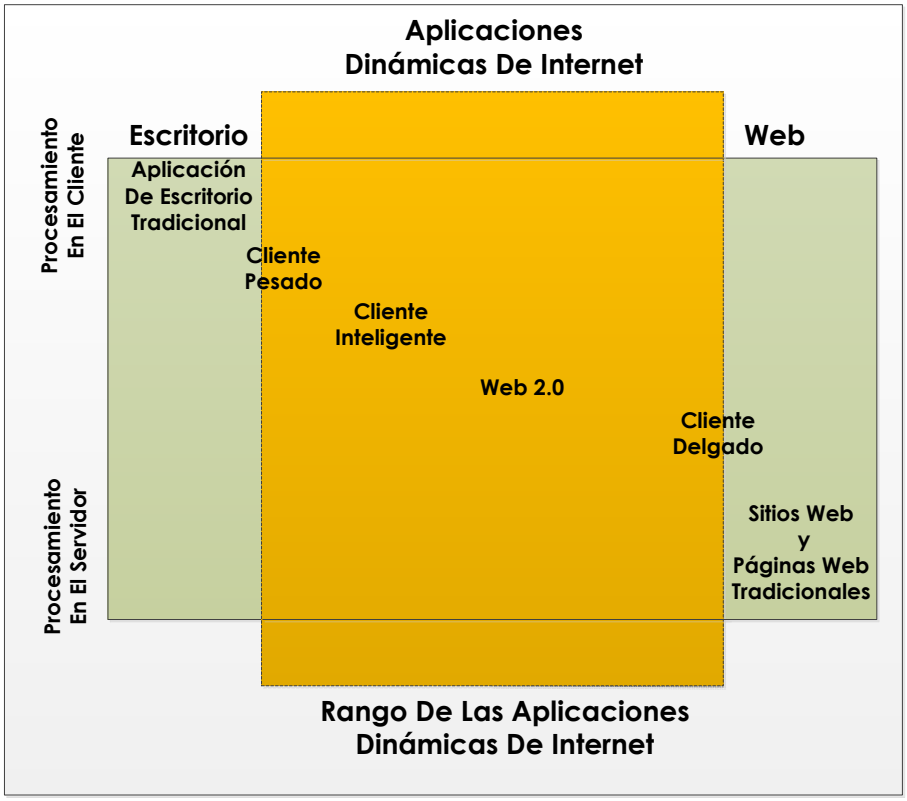


Figura 2. 8 Aplicaciones Dinámicas de Internet

2.7 Adobe Air

2.7.1 Concepto

AIR es una aplicación potente en tiempo de ejecución que permite a los desarrolladores crear aplicaciones de escritorio multiplataforma en múltiples entornos de desarrollo: Flex, Flash y JavaScript. Las aplicaciones AIR tienen todas las características que se esperaría de una aplicación Web, pero además, proporcionan funcionalidades que esperaría de una aplicación de escritorio, incluyendo el acceso al

sistema de archivo, monitorización de la red y una de base de datos incorporada.

Adobe AIR, cuyo nombre clave es Apollo 1 es un entorno de ejecución multiplataforma para la construcción de aplicaciones RIA (Rich Internet Applications) utilizando Adobe Flash, Adobe Flex, HTML y AJAX, las cuales como ya se mencionó pueden usarse como aplicaciones de escritorio.

El 19 de marzo de 2007, Adobe liberó una versión preliminar de AIR (llamada Apolo) junto con un SDK(Software Development Kit) y una extensión para el desarrollo de aplicaciones Apolo con Adobe Flex. El 10 de junio de 2007, pasó a llamarse AIR (Adobe AIR) y se liberó una versión beta del entorno de ejecución. Una versión alfa de AIR para Linux fue publicada el 31 de marzo de 2008.

AIR intenta ser un entorno de ejecución versátil, ya que permite que el código Flash, HTML o JavaScript existentes sea reutilizado para construir programas tradicionales de escritorio. Adobe lo posiciona como un entorno de ejecución sin navegador para que aplicaciones ricas de Internet (RIA) se puedan desplegar en el escritorio, en lugar de un framework de aplicaciones. Las diferencias entre cada paradigma de despliegue proporciona a ambas ventajas y desventajas. Por ejemplo, una aplicación rica de Internet desplegada en un navegador no requiere instalación, mientras que una desplegada con AIR requiere que sea empaquetada, firmada digitalmente, e instalada en el sistema de archivos local de los usuarios. Sin embargo, esto último ofrece almacenamiento local ilimitado y acceso al sistema de archivos, mientras que las aplicaciones desplegadas en el navegador están limitadas por las restricciones del mismo, y donde los datos, por lo general, se eliminan periódicamente. Sin embargo, en la mayoría de los casos, las aplicaciones ricas de Internet almacenan los datos de los

usuarios en sus propios servidores, pero la capacidad para consumir y trabajar con datos en el sistema de archivo local de un usuario permite una mayor flexibilidad cuando una aplicación está trabajando sin conexión.

2.7.2 Aplicaciones

Las aplicaciones AIR pueden funcionar sin conexión a internet y, a continuación, activar aún más la funcionalidad de carga de datos o cuando la conexión esté disponible. Un ejemplo de ello es eBay Desktop, que permite a los vendedores completar una lista fuera de línea y luego subirlo cuando están conectados a Internet.

Otras compañías usan actualmente AIR incluyendo AOL para el servicio de Top 100 Music Videos, NASDAQ Market Replay así como finetune para sus aplicaciones de escritorio.

2.7.3 Entorno de Desarrollo

Actualmente, Adobe ofrece tres formas de desarrollar aplicaciones AIR:

1. HTML/AJAX, a través de Adobe Dreamweaver CS3 u otros programas de edición de HTML junto con AIR SDK.6

2. Adobe Flex Builder 3

3. Adobe Flash CS3

2.7.4 Opciones de Datos

AIR actualmente tiene cuatro maneras de trabajar con datos:

1. Servidor de base de datos a través de servicios Web.

2. Archivo local de XML.

3. Base local de datos de SQLite enviadas con AIR.

4. Almacenamiento de cifrado local incluido con AIR.

2.8. ColdFusion

Adobe ColdFusion es un servidor de aplicaciones y software que permite a los desarrolladores construir rápidamente, desplegar y mantener aplicaciones robustas de Internet para la empresa. ColdFusion, combina un lenguaje intuitivo, basado en tags, con herramientas visuales y un servidor de aplicaciones web confiable, para entregar la manera más rápida de desarrollar poderosas aplicaciones web.

ColdFusion es una herramienta que corre en forma concurrente con la mayoría de los servidores web de Windows, Linux y Solaris (también en servidores web personales en Windows 98 y puede ser usado para intranets). El servidor de aplicaciones web de ColdFusion trabaja con el servidor HTTP para procesar peticiones de páginas web. Cada vez que se solicita una página de ColdFusion, el servidor de aplicaciones ColdFusion ejecuta el script o programa contenido en la página.

ColdFusion es un lenguaje de programación, puede crear y modificar variables igual que en otros lenguajes de programación que nos son

familiares. Posee controles de flujo de programas, como IF, Switch Case, Loop, etc. Tiene muchas funciones built-in para realizar tareas más complicadas como averiguar qué día caerá el 3 de Agosto del 2012 "DayOfWeekAsString (DayOfWeek ('2012/08/03'))".

No es un lenguaje de bases de datos, pero interactúa de manera simple con bases de datos (Sybase, Oracle, MySQL,SQL, o Access). Usando SQL estándar, las páginas y aplicaciones web pueden fácilmente recuperar, guardar, formatear y presentar información dinámicamente. ColdFusion es un lenguaje basado en tags, si se ha trabajado con HTML, será fácil utilizar CFML (ColdFusion Markup Language). Muchas de las funciones poderosas de ColdFusion, como leer desde y escribir en discos duros del servidor, son basadas en tags. Así como el tag <Table> puede tener argumentos como 'width' o 'align', el tag <CFFILE> tiene argumentos que especifican 'action=read/write/copy/delete', path=' etc.

El tag <CFFORM> construirá automáticamente todo el código JavaScript para verificar los campos requeridos antes de hacer submit al form. ColdFusion también tiene tags para COM, Corba y Applets y Servlets de Java.

2.8.1 Características

Es escalable: ColdFusion fue diseñado para desarrollar sitios complejos y de alto tráfico. A veces, el problema más grande para un diseñador web es que su sitio se vuelve popular. ColdFusion está diseñado para correr en máquinas multi-procesador, y permite construir sitios que pueden correr en clusters de servidores.

Es un lenguaje server-side: A diferencia de JavaScript y Applets Java, que corren en el cliente o en browsers, ColdFusion corre en el servidor web. Esto significa que los scripts escritos en ColdFusion correrán de la misma manera en cualquier browser.

Gestor de servidores: Ahorra tiempo realizando tareas administrativas en varios servidores de forma simultánea desde una consola central. Puede crear fuentes de datos, programar tareas, aplicar revisiones, borrar cachés y comparar la configuración entre varios servidores ColdFusion

2.8.2. Ambiente De Programación

ColdFusion soporta un poderoso lenguaje de scripting en el lado del servidor, ColdFusion Markup Language (CFML), que es extremadamente fácil de aprender y se integra limpiamente con todos los lenguajes y tecnologías web populares. ColdFusion trabaja con múltiples arquitecturas a través de la integración de COM, CORBA y EJB. También puede ser fácilmente extendido con nuevos componentes creados con Java Servlets, clases Java, o C/C++.

- Mejora la productividad gracias al lenguaje de scripting del servidor basado en tags, aplicado de manera única en aplicaciones web.
- Acelera el desarrollo con un conjunto poderoso de herramientas poderosas de diseño, programación, depuración e implantación.
- Permite a los equipos de desarrollo colaborar de manera más efectiva compartiendo el mismo servidor y trabajando local o remotamente.

2.8.2.1. Ensambla soluciones poderosas fácilmente:

- El servidor ColdFusion provee funcionalidades built-in como graficar, seguridad y búsqueda.
- Integración completa con la empresa, se conecta con todo el rango de sistemas backend, incluyendo bases de datos, servidores de mail, directorios, y aplicaciones empaquetadas. Se integra con tecnologías de empresa y

de internet, incluyendo COM, CORBA, EJB, XML, C/C++ y Java.

- Posee inteligencia de negocios. Permite crear planillas y reportes tabulares de calidad profesional.
- Completa búsqueda de texto. Permite indexar fácilmente y buscar muchos tipos de contenido, incluyendo páginas web y documentos Microsoft Office 2000.

2.8.2.2. Entrega un alto desempeño y confiabilidad:

- **Arquitectura de alto desempeño.** Asegura que las aplicaciones sean de implantación multiplataforma, entrega un avanzado thread pooling, caching de páginas built-in, consultas persistentes y pooling de conexiones a bases de datos.
- **Administración fácil**. Simplifica la implantación y la administración del servidor a través de una poderosa consola de administración basada en web, reportes robustos de servidor y herramientas de análisis, además de integración con los sistemas de administración de la empresa.
- **Clustering de servidor**. Provee balance de carga y recuperación automática para asegurar que las aplicaciones se mantengan consistentemente disponibles y se escala fácilmente para manejar tráfico creciente.

2.8.2.3. Ventajas

CFML hace fácil la programación web para nuevos desarrolladores, con más de 70 tags CFML y sobre 200 funciones personalizadas, prácticamente cualquier aplicación web puede ser construida rápidamente. ColdFusion puede ser usado en un sitio cada vez que se necesita interacción con el usuario.

Procesa formularios, hace seguras algunas partes del sitio, y recolecta o publica datos. Se puede usar para construir diarios murales, clientes de POP mail, calendarios en línea, y salas de chat. Se pueden escribir scripts para rastrear estadísticas.

ColdFusion es muy útil en el área de mantenimiento sobre otras herramientas middleware para crear sitios web dinámicos, ya que:

- ✓ Esconde la complejidad, usa menos líneas y son más intuitivas para alcanzar resultados, permite al usuario migrar a otros servidores web y motores de bases de datos con pocos cambios y sin plugins externos.

- ✓ ColdFusion simplifica el almacenamiento de variables, el programador puede manipular fácilmente las variables apropiadas a su sesión lógica. Lo mismo pasa con las variables en el servidor, en la aplicación y al nivel de página. Otros middleware necesitan más compromiso del programador y más trabajo para que sea escalable.

- ✓ ColdFusion genera y envía JavaScript transparentemente on the fly cuando ciertos tags de input son utilizados. Esto facilita el chequeo de inputs del lado del cliente sin forzar al programador a escribir, revisar y modificar JavaScript para hacer esto.

- ✓ ColdFusion tiene un buen manejo de errores y depuración. Permite redirigir la información detallada para la depuración a direcciones IP que el programador provea. Cuando un motor de bases de datos arroja un error, ColdFusion sugiere causas posibles. Le permite al programador crear sus propios manejadores de errores cuando se necesita cuidado

especial. Con ColdFusion se pueden personalizar los mensajes de error para situaciones específicas y puede proveer un nivel de detalle para los usuarios y uno diferente para los mantenedores.

- ✓ Las aplicaciones en ColdFusion pueden cambiar de plataformas y motores de bases de datos. Se pueden cambiar las aplicaciones de ColdFusion a diferentes sistemas operativos y servidores web con pequeños cambios, y se puede incluso cambiar los motores de bases de datos con un poco más de esfuerzo.

2.8.2.4. Ventajas inigualables de ColdFusion

ColdFusion viene con habilidades que otros middleware no pueden alcanzar sin necesitar agregados. Viene con un motor para indexar sitios web. Se puede realizar balance de carga dinámico, se pueden mantener porciones de código como propietarias. Permite agregar componentes de servidores de otros lenguajes, puede usar COM, CORBA y objetos JavaBeans creados por otras herramientas. Se pueden hacer consultas persistentes para mayor velocidad. Se integra con el monitor de desempeño y el monitor de seguridad de NT. Se puede poner a los usuarios en una "caja de arena" (por ejemplo, para soportar múltiples sitios web en un sólo host). Se puede modificar el registro. Permite continuar usando scripts CGI existentes. El código ColdFusion puede ser generado on the fly con herramientas que vienen con su usual editor, ColdFusion Studio. El lenguaje es extensible, convierte datos hacia y desde XML y permite conectar un sitio con otros como en un browser para extender el alcance de la aplicación.

2.9 Web Services

Actualmente en el desarrollo Web se ha centrado en una gran cantidad de tecnologías y muchas de ellas incompatibles entre sí, por lo cual sabemos que las arquitecturas basadas en tecnología de componentes está tomando un papel principal dentro del desarrollo Web.

Los Web Services se proponen como una alternativa para facilitar la intercomunicación entre diferentes arquitecturas de componentes, ofreciendo una visión de dichas arquitecturas basada en servicios.

Permiten la comunicación entre aplicaciones o componentes de aplicaciones de forma estándar a través de protocolos comunes, como http, y de manera independiente al lenguaje de programación, plataforma de implantación, formato de presentación o sistema operativo. Un Web service es un contenedor que encapsula funciones específicas y hace que estas funciones puedan ser utilizadas en otros servidores.

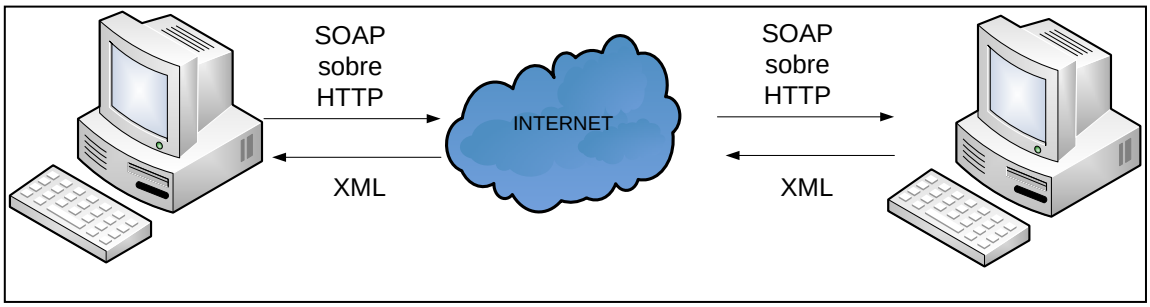


Figura 2. 9 Web Services

Algunas ventajas que presentan los Web services son

- Son programables
- Están basados en XML, que es un lenguaje abierto
- Son auto descriptivos
- Pueden buscar registros de otros Web services

2.9.1. Beneficios de Web services sobre otras tecnologías

- Escaso acoplamiento. El cliente no necesita conocer nada acerca de la implementación del servicio al que está accediendo, salvo la definición WSDL.
- Independencia del lenguaje de programación. El servidor y el cliente no necesitan estar escritos en el mismo lenguaje
- Independencia del modo de transporte. SOAP puede funcionar sobre múltiples protocolos de transporte, como por ejemplo HTTP, HTTPS, HTTP-R, BEEP, JABBER, IIOP, SMTP o FTP.
- Múltiples modos de invocación. Los servicios web soportan tanto invocación estática como invocación dinámica.
- Múltiples estilos de comunicación. Los servicios web soportan tanto comunicación síncrona (RPC) como comunicación asíncrona (mensajería).
- Extensibilidad. Al estar basados en XML, los servicios web son fáciles de adaptar, extender y personalizar.

2.9.2 Estándares empleados

- ***Web Services Protocol Stack:*** Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- ***XML (Extensible Markup Language):*** Es el formato estándar para los datos que se vayan a intercambiar.
- ***SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call):*** Protocolos sobre los que se establece el intercambio.
- ***Otros protocolos:*** los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (HyperText Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- ***WSDL (Web Services Description Language):*** Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

- ***UDDI (Universal Description, Discovery and Integration):*** Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.
- ***WS-Security (Web Service Security):*** Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

2.9.3 Razones para crear servicios Web

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante *firewalls* -que filtran y bloquean gran parte del tráfico de Internet-, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran *ad hoc* y poco conocidas, tales como EDI (Electronic Data Interchange), RPC (Remote Procedure Call), u otras APIs.

Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.

Ejemplos De Código:

Consulta a una base de datos:

```
<cfquery name="nombredelaconsulta"
datasource="conexion_odbc">
  SELECT *
  FROM table
  WHERE campo = 'hola'
</cfquery>
```

Mostrar la respuesta de una consulta:

```
<cfoutput query="nombredelaconsulta">
   #nombredelaconsulta.campo#
   <!---Las variables se escriben entre # #. Este texto es un
comentario --->
</cfoutput>
```

En conclusión un Web Service es cualquier servicio que pueda llamarse mediante los protocolos utilizados en el World Wide Web por un cliente situado remotamente.

Lo importante de los web services es el modo en el que se llama y devuelve la respuesta. Toda la comunicación entre un web services y su cliente e basa en XML, y está por tanto estandarizado, el cliente envía una petición en XML al web service y como respuesta también recibirá un XML, ambos XMLs se han creado bajo el estándar SOAP (Simple Object Access Protocol). Por lo que carece de importancia el lenguaje de programación que tenga el cliente así como el lenguaje de programación en el que este desarrollado el web service.

Los métodos disponibles en el web service, los parámetros que recibe así como el tipo del valor que devuelve también se definen en un XML estándar, el WSDL (Web Service Description Language). Un XML WSDL puede interpretarse por cualquier programa realizado en cualquier lenguaje y así escribir el mensaje SOAP a elaborar con el que trabajará.

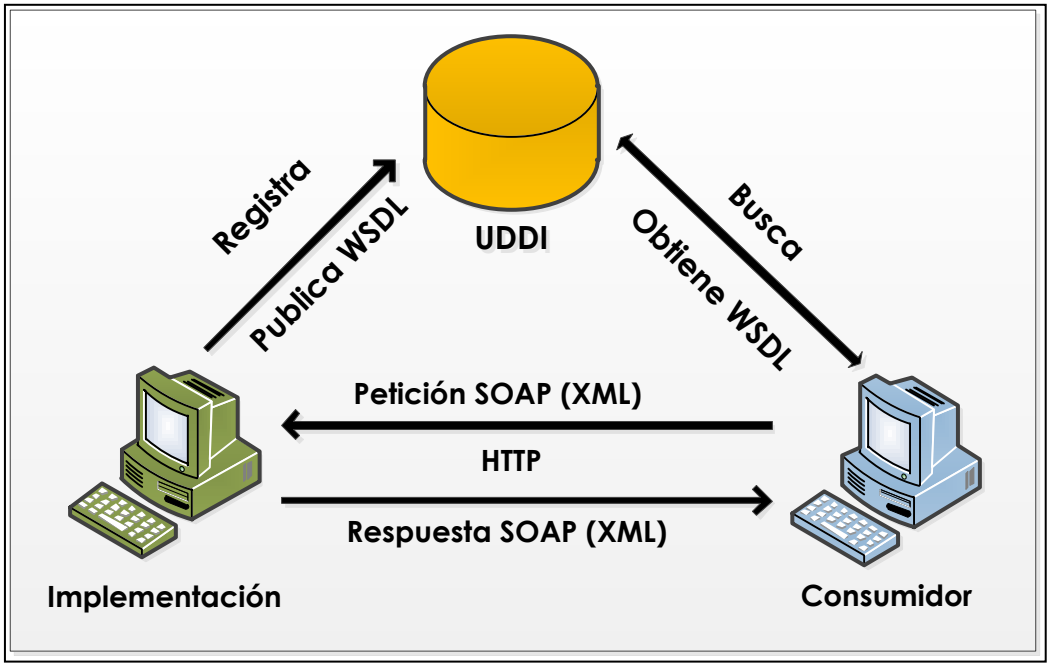


Figura 2. 10 Modo en el que se llama y devuelve la respuesta un Web Service

2.10. ColdFusion y Web Services

Internet ha cooperado a facilitar a las empresas con una mejor comunicación interna y entre empresas, proporcionando un rápido acceso a la información. Sin embargo, la navegación por páginas basadas en datos no responde oportunamente a las necesidades de negocio de muchas empresas. Sería mejor poder disponer de sitios Web programables que vincularan directamente organizaciones, aplicaciones y servicios; los web services XML son este vínculo directo entre las aplicaciones. Al enlazar nuestras aplicaciones y sitios Web a servicios Web XML con ColdFusion,

tenemos la oportunidad de expandir la funcionalidad que ofrece nuestro sitio Web a los usuarios.

ColdFusion y sus componentes se basan en OOP, pero no en su totalidad; tienen sus métodos y construcciones que proveen de código reutilizable. Las funciones y métodos de ColdFusion en Web Services son a través de métodos CFC.

Desde sus inicios, Internet ha permitido a la gente acceder a contenidos almacenados en equipos remotos. Este contenido puede ser estático, como un documento representado por un archivo HTML, o dinámicos, como el contenido de una página de ColdFusion.
Los servicios Web permiten el acceso a las aplicaciones que alguien más ha creado y ha puesto a disposición en un equipo remoto. Con un servicio web, se puede hacer una petición a una aplicación a distancia para realizar una acción.

Por ejemplo, se puede solicitar una cotización, pasar una cadena de texto para traducir, o solicitar información de un catálogo de productos. La ventaja de los servicios web es que no tenemos que volver a crear la lógica de la aplicación que alguien ya creó, por lo tanto, podemos construir nuestras aplicaciones con mayor rapidez
Hacer referencia a un servicio Web remoto dentro de una aplicación de ColdFusion se llama consumir servicios web. Dado que los servicios web, se adhieren a una interfaz estándar, independientemente de la tecnología de la aplicación, se puede consumir un servicio web en ejecución como parte de una aplicación ColdFusion, o como parte de una red o aplicación Java.
También podemos crear nuestros propios servicios web y ponerlos a disposición de los demás para el acceso remoto, llamado publicación de servicio web. Las aplicaciones que consumen el servicio web pueden ser aplicadas en ColdFusion o por cualquier aplicación que reconozca el estándar de servicios web.

Un acceso a un servicio Web es similar a la llamada de una función. En lugar de que la función haga a la llamada de una biblioteca en su equipo, hace referencia a su funcionalidad remota a través de Internet.

Una de las características de los servicios web es que son auto-descriptibles. Esto significa que una persona que hace un servicio web disponible, también publica una descripción de la API para el servicio Web como un *Web Services Description Language (WSDL)*.

Un archivo WSDL es un documento con formato XML que incluye información sobre el servicio web, incluyendo la siguiente información:

- Las operaciones que se puede llamar al servicio web
- Los parámetros de entrada que se pasa a cada operación
- Retornar valores de una operación

Pasos para consumir el web service:

1. Analizar el archivo WSDL del servicio web para determinar su interfaz.

   Un servicio web, hace que su archivo WSDL disponible esté asociado a través de Internet. Se debe conocer la dirección URL del archivo WSDL que define el servicio.

2. Hacer una petición al servicio web.

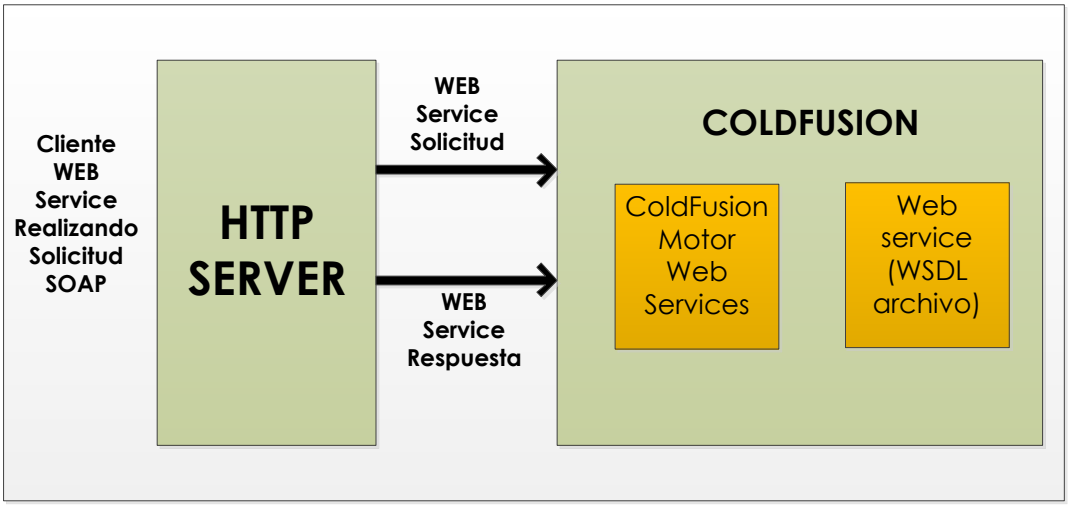


Figura 2.11 Implementación de ColdFusion Con Web service

2.10.1. Crear Un Web Service Con ColdFusion

Primero se creará una función, la cual llamaremos ejemplo.cfm. El siguiente código muestra cómo se genera una cadena después de aceptar una cadena diferente.

El argumento es enviado a través de **"argumentos. Nombre"** que es "Juan" en este caso.

```
<cffunction name="ejemplo" returntype="string">
<cfargument name="nombre" type="string" required="yes">
name="nombre" <cfargument tipo="String" required="yes">
<cfreturn "<h3><font color='blue'>Bienvenidos  A Mi Sitio De Web
Service</font>
    </h3> " & arguments. name &"! </ H3> "& argumentos. Nombre
" & " ¿Qué te gustaría hacer? ">

</cffunction>

<cfoutput>  # ejemplo ("Juan")# </cfoutput>
```

Pantalla de ejemplo.cfm

Figura 2.12 Pantalla ejemplo.cfm

Ahora cambiaremos la función ejemplo.cfm en un web service, mediante el uso de las etiquetas CFC, la cual ahora llamaremos ejemplo.cfc.

```
<cfcomponent>
        <cffunction name="ejemplo" access="remote"
returntype="string"
        output="no">
        <cfargument name="nombre" type="string"
required="yes">
<cfreturn "<h3><font color='blue'> Bienvenidos  A Mi Sitio De
Web Service </font>
        </h3> " & arguments.nombre &"! " & "¿Qué te gustaría
hacer? "">
        </cffunction>
</cfcomponent>
```

En primer lugar, observamos que el código de la función se anida entre las etiquetas ***cfcomponent***. También se observa que un par atributos más se agregan a la función. Al servicio web se accede de

forma remota, por lo que el acceso está configurado como "**remote**". Dado que es un servicio que no puede acceder a él directamente para una salida, lo hemos fijado "no" en "**output**". Cuando el consumidor lo llama, este servicio se pone disponible después de que recibe el mensaje desde el servidor. La etiqueta **cfreturn** también es necesaria, porque esto es lo que devuelve el servicio.

2.10.2. Consumir el servicio web

Para saber si este web service está funcionando crearemos un cliente que consuma este servicio.

Escribir el código de cliente no es diferente a escribir una página de CFM, se usa etiquetas especiales, como "cfinvoke" la cual llama al servicio, y para llamarlo utilizaremos el WSDL. Cuando se invoca el servicio, lo invoca con un argumento, en este caso "Sr. Pérez". La variable de retorno es una cadena que se muestra como lo haría normalmente con las etiquetas "**cfoutput**".

```
<cfinvoke
        webservice="http://localhost/c_fusion/webservice_ejemplo.cfc?wsdl"

        method="ejemplo"

        returnvariable="strg">

<cfinvokeargument name="name" value="Sr. Pérez"/>
</cfinvoke>
<cfoutput>
#strg#
</cfoutput>
```

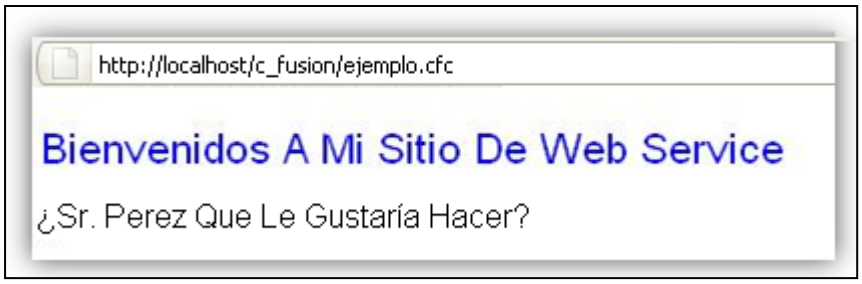


Figura 2.13 Pantalla ejemplo.cfc

Es sencillo crear un servicio web en ColdFusion y codificar un consumidor es igual de sencillo, todo lo que se necesita es proporcionar una referencia WSDL, una variable de retorno, y los argumentos necesarios que necesita el servicio. Aunque la función utilizada es muy simple, los tipos de datos más complejos, así como los datos desde el servidor se pueden proporcionar desde el web service. La clave es la interoperabilidad, el cliente consumidor puede utilizar cualquier otro lenguaje de programación.

2.11 Conclusión

Durante el desarrollo de este capítulo hemos definido que son las aplicaciones de Internet Enriquecidas (RIA´s), las cuales nos permiten mejorar el desempeño y la presentación visual de las aplicaciones construidas, disminuyendo el tiempo de desarrollo y añadiendo la posibilidad de tener la misma aplicación tanto en la Web como en el escritorio del cliente. Otro punto importante a considerar es la mejora en cuanto al tráfico que se genera ya que con esta tecnología no se requiere de recargas continuas para actualizar los datos de pantalla ya que la aplicación hace pedidos específicos al servidor sin la necesidad de recargar en cada momento el contenido, reduciendo tanto el tiempo de espera del usuario como el tráfico de información generado por la recarga y además teniendo la posibilidad de integrar un feedback para indicar al usuario de que está pasando en la aplicación en cada momento.

Otro gran beneficio de las RIA´s es el uso de controles avanzados como Drag and Drop en el navegador ya que antes se podía usar solo en aplicaciones de escritorio.

Además en este capítulo hemos identificado las diferentes herramientas con las que podemos desarrollar e interactuar entre aplicaciones RIA´s incluyendo los Web Services, Adobe AIR, Cold Fusion entre otros y hemos podido conocer la forma de clasificar esta nueva tecnología.

CAPÍTULO III

CAPÍTULO 3

PLANEACIÓN Y ANÁLISIS DEL SISTEMA DE COMERCIO ELECTRÓNICO DE ENTREGAS Y REPARTOS "QUICK DELIVERY"

3.1 Introducción

En este capítulo realizaremos el análisis de nuestro sistema basándonos en la resolución de los problemas y necesidades que hemos identificado previamente. Para poder realizar la codificación del sistema debemos conocer el dominio total de la información de un problema, para con esto definir las posibles funciones que ayudaran a sobrellevar estas necesidades.

Antes de realizar el análisis del sistema que estamos desarrollar hemos decidido llevar a cabo una entrevista de campo para conocer de boca de los usuarios sus necesidades y dificultades a la hora de procesar un pedido, procurando dar mayor énfasis a aquellas cosas que los usuarios califican como indispensables para poder realizar su trabajo.

Mediante el desarrollo de estos modelos mostraremos la funcionalidad que va a tener nuestro sistema así como su interacción con el usuario, como se estructura cada parte del sistema contemplando también el tratamiento que el sistema debe dar a los datos y como se almacenaran estos en la base de datos. Además identificaremos los diferentes usuarios que interactúan con el sistema y la función que cada uno debe de cumplir.

3.2 Delivery

La gestión del Delivery se preocupa del diseño, planificación, implementación y mejoramiento de los flujos asociados a la entrega, generalmente sujeta a restricciones de tiempos y costos. El resultado de la gestión del reparto o Delivery es la operación de colocar en el tiempo acordado, en las condiciones acordadas, y a la persona adecuada la cantidad precisa de un bien o servicio adquirido.

El avance en las comunicaciones y el uso de las nuevas tecnologías de la información han permitido que también sea una condición del servicio disponer de trazabilidad del producto en tiempo real.

El reparto o Delivery adquiere distintos grados de complejidad dependiendo de la industria y mercados donde se aplica, así, otra propiedad del reparto o Delivery es la posibilidad de maximizar la interacción humana al final del flujo, con posibilidades de emplearse en la fidelización de clientes (Envío de Regalos), estudios de mercado (Envío de Encuestas) u otros fines.

Delivery se encarga de resolver todos los traslados entre un lugar de origen (Almacén, Sucursal, Centro de distribución, etc) y el cliente o usuario final.

En una secuencia ideal, el flujo del reparto se inicia desde que el usuario o cliente finaliza el ingreso de su solicitud o requerimiento mediante un portal de atención remota, sea este una página web, un software específico diseñado para estos fines, un centro de llamados o incluso en atención presencial donde solicita el reparto a domicilio y finaliza cuando el solicitante recibe los bienes o servicios enviados.

Por todas estas razones se ha pensado en construir un sitio web que distribuya varios tipos de productos comestibles, con el fin de que el usuario pueda realizar su pedido las 24 horas, los 7 días de la semana y que sea entregado en cualquier parte de la ciudad. Para el desarrollo de este sitio

utilizaremos nuevas herramientas de desarrollo como son las llamadas tecnologías RIA.

3.3 Entrevista

Para comprender mejor el funcionamiento del servicio de entregas a domicilio se realizó una entrevista a una pequeña empresa la cual se dedica a vender productos vía celular y chat; así podemos tener conocimiento más a fondo de los requisitos que necesitaremos para el desarrollo de nuestra página web.

Anexo 1 (Entrevista)

3.4. Proceso del desarrollo web

El proceso a utilizarse es el de Programación Extrema (XP), este se basa en la simplicidad, la comunicación y la retroalimentación continua de código. El objetivo es aumentar la productividad al desarrollar el software.

La programación extrema utiliza un enfoque orientado a objetos como paradigma de programación preferido.

En la siguiente figura podemos observar el ciclo de vida típico que utiliza esta metodología:

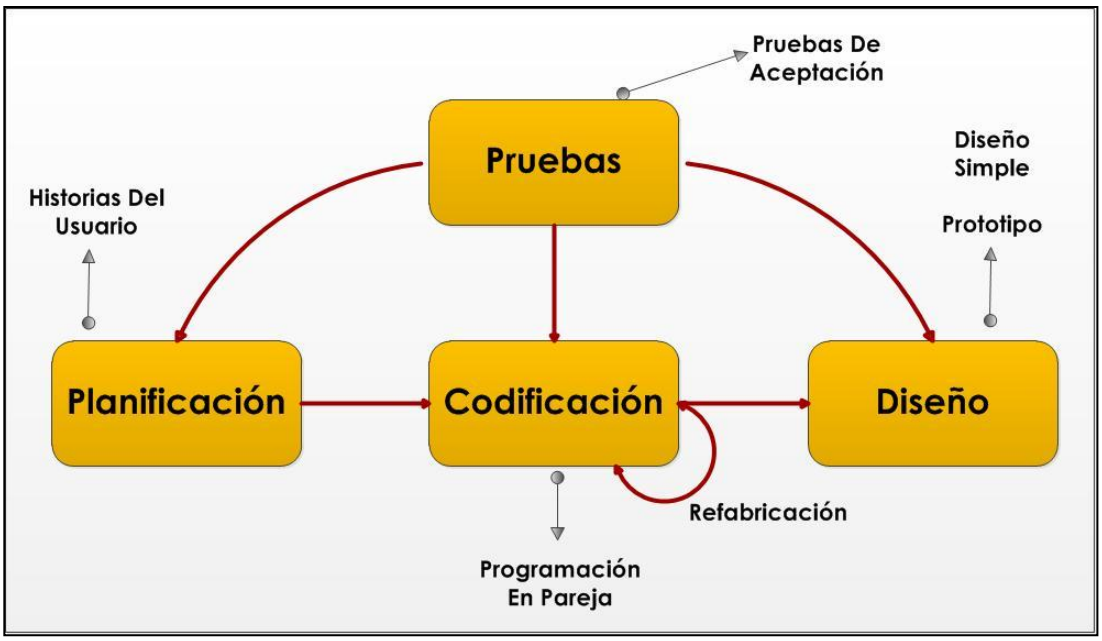


Figura 3.1 Ciclo De Vida Programación Extrema

- **Planificación:**

Es la primera actividad en el proceso de desarrollo. Comienza creando una serie de historias de usuarios (similares a los casos de uso) que describen la funcionalidad del software que se va a construir. El cliente les asigna una prioridad y el equipo de desarrollo evalúa cada una y le asigna un periodo de desarrollo. Si la historia supera más de tres semanas de desarrollo se divide la historia en historias menores.

Una vez establecido un acuerdo detallando la fecha de entrega, el equipo de desarrollo ordena las historias para implementar antes las que tengan mayor prioridad.

Conforme avanza el trabajo de desarrollo, el cliente puede agregar nuevas historias con nueva funcionalidad, de esta manera, la programación extrema es una metodología que acepta fácilmente requisitos cambiantes durante el desarrollo de software.

- **Diseño:**

El diseño en la programación extrema sigue el principio de hacerlo todo simple. El diseño se va modificando a lo largo de todo el proceso de desarrollo.

- **Codificación:**

La programación extrema recomiendo que después de diseñar las historias el equipo no se debe comenzar la codificación sino que debe desarrollar una serie de pruebas de unidad que les ayuden a centrarse en lo que debe implementase para pasar esa prueba.

Un concepto clave en esta etapa es la programación en pareja de tal forma que dos personas trabajan juntas en un ordenador para crear el código de una historia siguiendo un estándar de codificación. Este enfoque asegura la calidad del código, ya que dos cabezas piensan mejor que una, además permite la rápida comunicación entre las dos personas y un mejor conocimiento del problema que se quiere solucionar.

- **Pruebas:**

Las pruebas de unidad creadas deben ser automatizadas para que puedan ejecutarse de manera fácil y rápida. De esta forma podemos modificar el código y asegurarnos que funciona pese a los cambios producidos.

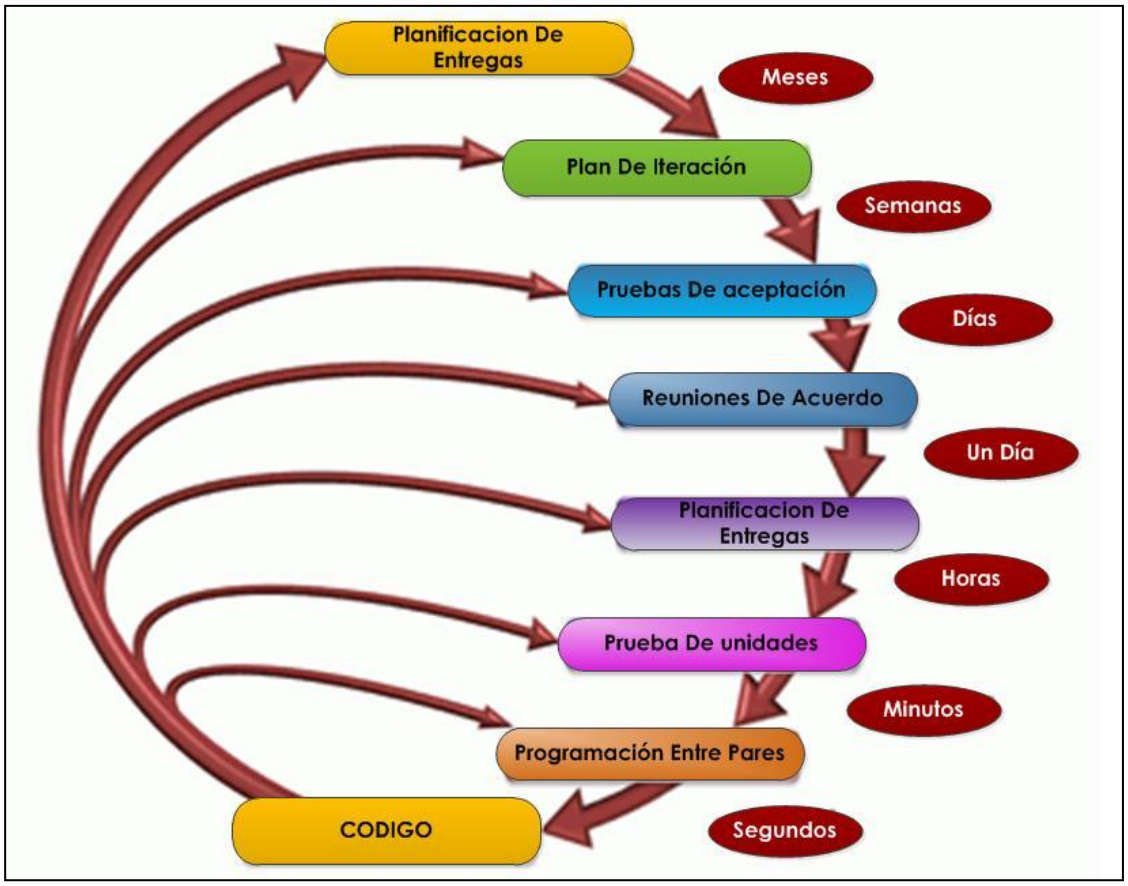


Figura 3. 2 Ciclo de retroalimentación y procesos de desarrollo en planificación

Se puede describir a la programación extrema de la siguiente manera:

- ✓ **Planificación incremental:** Los requerimientos se registran en tarjetas de historias. Los desarrolladores dividen estas historias en Tareas de desarrollo.

- ✓ **Entregas pequeñas:** Primero se desarrolla una versión con la funcionalidad más importante. Después se van añadiendo funcionalidades a las distintas entregas.

- ✓ **Diseño sencillo:** Solo se lleva a cabo el diseño necesario para cumplir con los requerimientos actuales.

- ✓ **Desarrollo previamente probado:** Se utilizan pruebas automatizadas antes de escribir código.

- ✓ **Programación en parejas:** Las parejas de desarrolladores trabajan en todas las áreas del sistema.

- ✓ **Integración continua:** Cuando acaba el trabajo de una historia se integra en el sistema entero. después de la integración se deben pasar todas las pruebas de unidad.

- ✓ **Cliente presente:** El cliente debe estar disponible ya que es miembro del equipo de desarrollo y es responsable de formular los requerimientos del sistema para que se puedan implementar.

3.5. Plan de trabajo

Hemos implementado un plan de trabajo para el análisis y desarrollo de nuestro sistema, el mismo que ha sido realizado en un diagrama de Gantt.

La finalidad de este esquema es el poder llevar a cabo un plan de tareas que se ajuste a los tiempos y recursos requeridos, y poder llevar un cronograma claro de lo que se va a realizar paso a paso en el proceso de esta aplicación web basada en tecnologías RIA.

A continuación presentamos el diagrama de Gantt con las tareas, fechas y tiempo de ejecución de las mismas.

Nombre	Fecha de inicio	Fecha de fin	Duración
Aplicación Web Quick Delivery	2/06/10	7/07/11	286
Recopilación De Información Teórica	2/06/10	26/06/10	18
Entrevista	7/07/10	8/07/10	1
Proceso Del Proyecto	9/07/10	10/07/10	1
Formulación Y Planeación	26/07/10	31/08/10	26
Recopilación De Requisitos	26/07/10	31/08/10	26
Requisitos De Contenido	26/07/10	5/08/10	8
Requisitos Funcionales	6/08/10	17/08/10	7
Requisitos No funcionales	18/08/10	26/08/10	6
Jerarquía De Usuarios	27/08/10	31/08/10	2
Análisis Del Proyecto	1/09/10	16/10/10	33
Modelo De Contenido	1/09/10	9/09/10	6
Modelo De Interacción	9/09/10	18/09/10	7
Modelo Funcional	20/09/10	24/09/10	4
Modelo De Configuración	27/09/10	1/10/10	4
Análisis Relación-Navegación	1/10/10	5/10/10	2
Modelo De Datos	6/10/10	16/10/10	8
Diseño Del Proyecto	21/10/10	4/12/10	32
Diseño De La Interfaz	21/10/10	5/11/10	11
Diseño Estético	8/11/10	17/11/10	7
Diseño De Contenido	18/11/10	25/11/10	5
Diseño Arquitectónico	26/11/10	4/12/10	6
Arquitectura De Contenido	26/11/10	1/12/10	3
Arquitectura De La WebApp	2/12/10	4/12/10	2
Gestión De Riesgos	6/12/10	10/12/10	4
Codificación	13/12/10	26/05/11	118
Desarrollo QDelivery	13/12/10	29/01/11	35
Desarrollo QMantenimiento	1/02/11	10/03/11	27
Desarrollo QEstadística	10/03/11	22/04/11	31
Desarrollo Aplicaciones Air Des...	25/04/11	14/05/11	15
Implementación De APIS	16/05/11	21/05/11	5
Implementacion De Librerías P...	24/05/11	26/05/11	2
Pruebas De La Aplicación Web	13/06/11	7/07/11	18

Figura 3.3 Diagrama De Gantt Parte 1

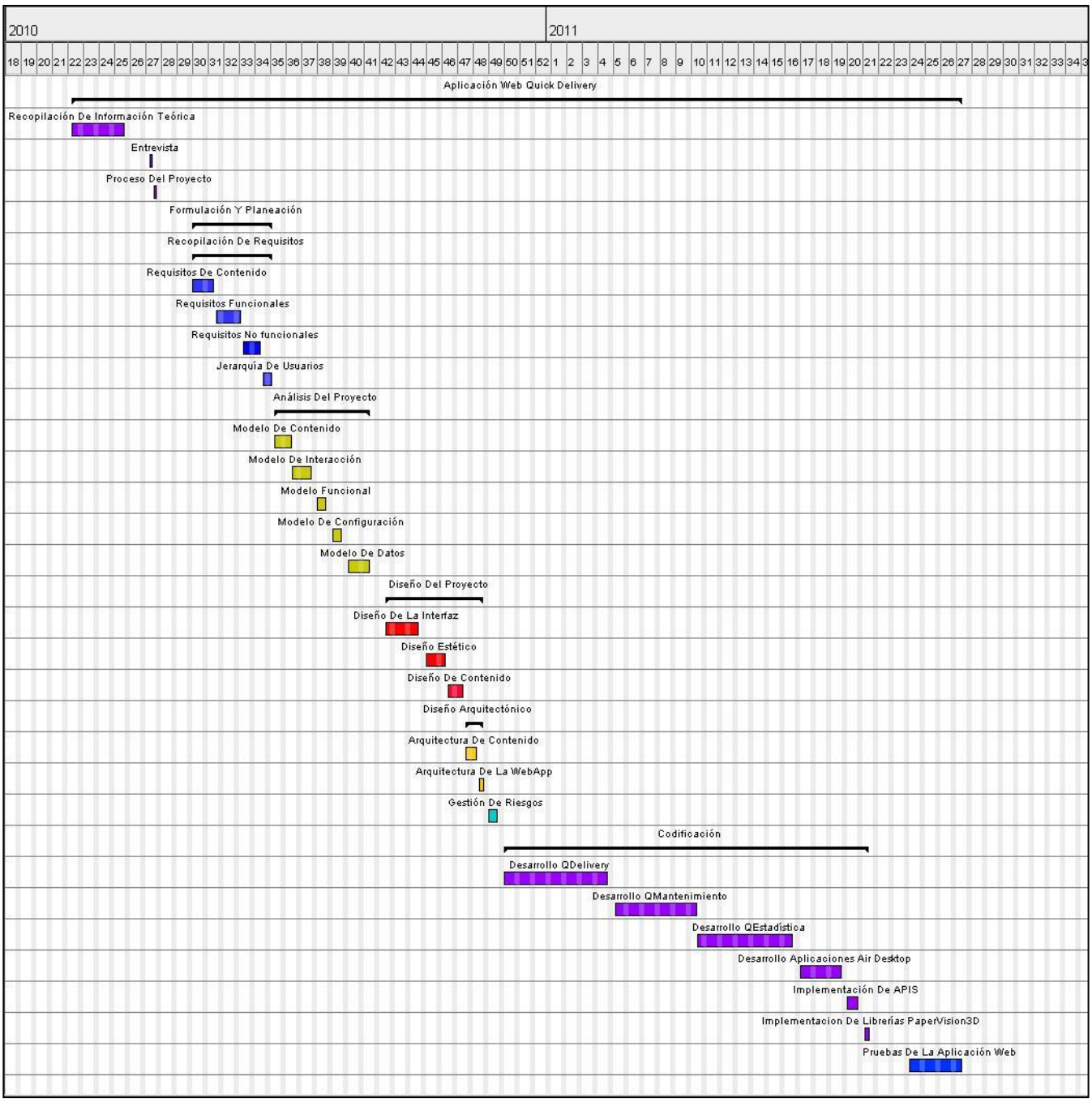


Figura 3. 4 Diagrama De Gantt Parte 2

3.6. Requisitos para el análisis de la aplicación web

Dentro del análisis se identificará el contenido que presentará la aplicación web para así identificar los requisitos específicos que se implementarán en el desarrollo, de modo que el ambiente y la infraestructura que apoyan la aplicación web puedan construirse.

3.6.1. Requisitos de contenido

- **Formularios:** El formulario 3D que se encuentra en el menú principal, contiene datos principales como nombre, dirección, correo electrónico y un comentario, el cuál será recibido por el administrador del sistema y contestado por el mismo.
  Además se contará con un formulario para los usuarios que deseen registrarse y quieran realizar compra en el sitio web, todos los datos ingresados serán validados.

- **Menús:** Existen distintos tipos de menú, el primero será un menú de navegación que estará en la cabecera de la página principal donde se encontrará la misión, visión, ayudas y un formulario en 3D para contactos, el segundo es un menú para el ingreso a las distintas aplicaciones que tendrá el sistema Quick Delivery; y el resto de aplicaciones contarán con menús fáciles de interactuar, y en ciertas ocasiones se podrá arrastrar y soltar productos hacia el carro de compras.

- **Cuentas:** Se realizará la creación de cuentas nuevas para los distintos usuarios, ya sean administrativos o simplemente usuarios que deseen comprar en el sitio web.

- **Carro de Compras**: Podremos ver los productos seleccionados, su valor, cantidad y total de la compra; y así mismo el usuario podrá guardar su compra para retornar luego a finalizar la compra.

- **Aplicaciones Desktop Air:** Se dispondrán de links para las descargas de las distintas aplicaciones de escritorio desarrolladas en air, y así el usuario pueda instalar en su computador y acceder al sitio sin tener que utilizar un navegador web.

- **Gráficos estadísticos:** Estos gráficos servirán para que el administrador del sitio pueda llevar un seguimiento de las compras realizadas en distintos rangos de fechas y por distintas categorías, para lo cual se implementarán gráficos de columnas, circular y de líneas.

- **Utilitarios:** Se añadirán distintas aplicaciones APIS de Google como mapas, traductor, entre otros, que el usuario podrá utilizar de forma rápida y sencilla desde el mismo sitio web de "Quick Delivery".

3.6.2. Requisitos funcionales

- El sistema permitirá comprar mediante el sitio web a los usuarios.

- Mostrar productos por categorías.

- El sistema debe registrar a los usuarios.

- El sistema permitirá almacenar productos.
- Buscar productos y su información.
- Guardar información de las compras realizadas.
- Formularios de usuarios y contactos.
- Manejo de carro de compras.
- Descargar aplicaciones Desktop Air.

3.6.3. Requisitos no funcionales

- Rendimiento de memoria.
- Compatibilidad con varios navegadores.
- Mantenimiento del sitio web.
- Versiones de las aplicaciones de Desktop Air.

3.6.4. Jerarquía de usuarios

Se ha organizado a todos los usuarios que intervienen en el sistema con el fin de entender mejor su rol en la aplicación web.

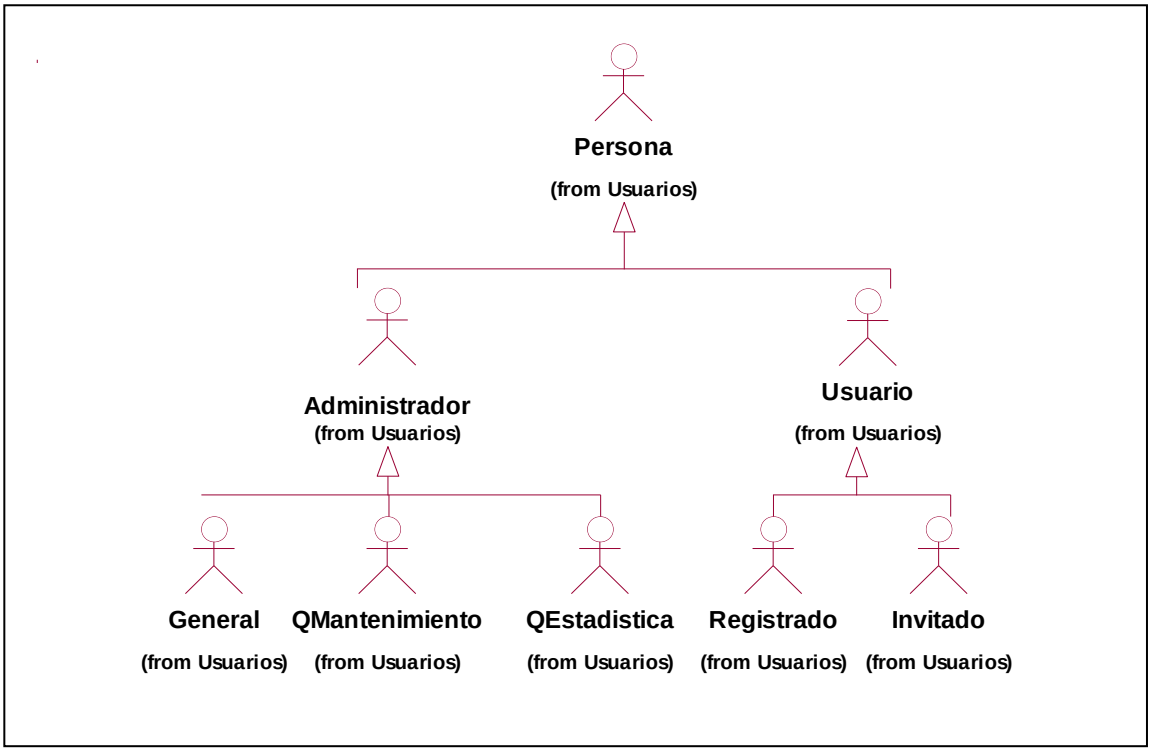


Figura 3. 5 Jerarquía De Usuarios

Usuarios Invitado: Un usuario visitante puede consultar e ir buscando entre las distintas categorías los productos disponibles. Esto permite al invitado guardar sus productos en un carro de compras temporal. Una vez que el visitante abandona el sitio web no desaparece el contenido de su carro de compras. Si el usuario visitante regresa y decide ingresar al sitio para crear una cuenta el este será añadido como usuario registrado automáticamente y sus productos seleccionados permanecerán en el carro de compras. Puede realizar comentarios en la página principal e ingresar a los distintos utilitarios disponibles y descargar la aplicación QDelivery Desktop.

Usuario Registrado: El usuario registrado o asociado puede añadir productos a su carro de compras, y volver más tarde para finalizar el pedido. Todos los productos permanecen en el carro de compras hasta que el asociado ha realizado el pedido, o hasta que sean eliminados del carro manualmente y finalmente este puede terminar la

compra. Igualmente podrá hacer comentarios o sugerencias en la página web e ingresar a los utilitarios y descargar la aplicación QDelivery Desktop.

Administrador: El administrador mantiene y administra el servidor y las bases de datos asociadas. Tiene la capacidad de crear y eliminar usuarios. Dentro de este usuario tenemos otro tipo de usuarios que de igual forma tienen usuario y contraseña para el ingreso al sistema:

- **Administrador de productos:** Este puede ingresar a la aplicación QMantenimiento, para el ingreso, modificación y eliminación de productos. Tiene la opción a descargar la aplicación QMantenimiento Desktop.

- **Administrador de estadísticas y ventas:** Este puede ingresar a la aplicación Qestadistica, para el seguimiento de las ventas de los productos. Tiene la opción a descargar la aplicación Qestadistica Desktop.

3.7. Análisis del Proyecto para la Aplicación Web

Para el análisis del proyecto se desarrollarán actividades de tal modo que aporten con modelos de análisis completo para el correcto desarrollo de la aplicación web.

Se realizará el siguiente análisis:

- El modelo de contenido
- El modelo de Interacción

- El modelo funcional
- El modelo de configuración
- Análisis relación-navegación

Los modelos es si mismos contienen elementos estructurales y dinámicos de la webApp. Los elementos estructurales son los que identifican las clases de análisis y los objetivos de contenido que se necesitan para la creación de un sitio web que cumpla con las necesidades del cliente. Los elementos dinámicos del modelo de análisis determinan cómo interactúan los elementos estructurales, entre ellos y con los usuarios finales.

3.7.1. Modelo de contenido

El modelo de contenido contiene los elementos estructurales que nos aportan una importante visión sobre el contenido de nuestra aplicación web, incluyendo entidades visibles para el usuario que se crean o utilizan a medida que éste interactúa con la aplicación web.

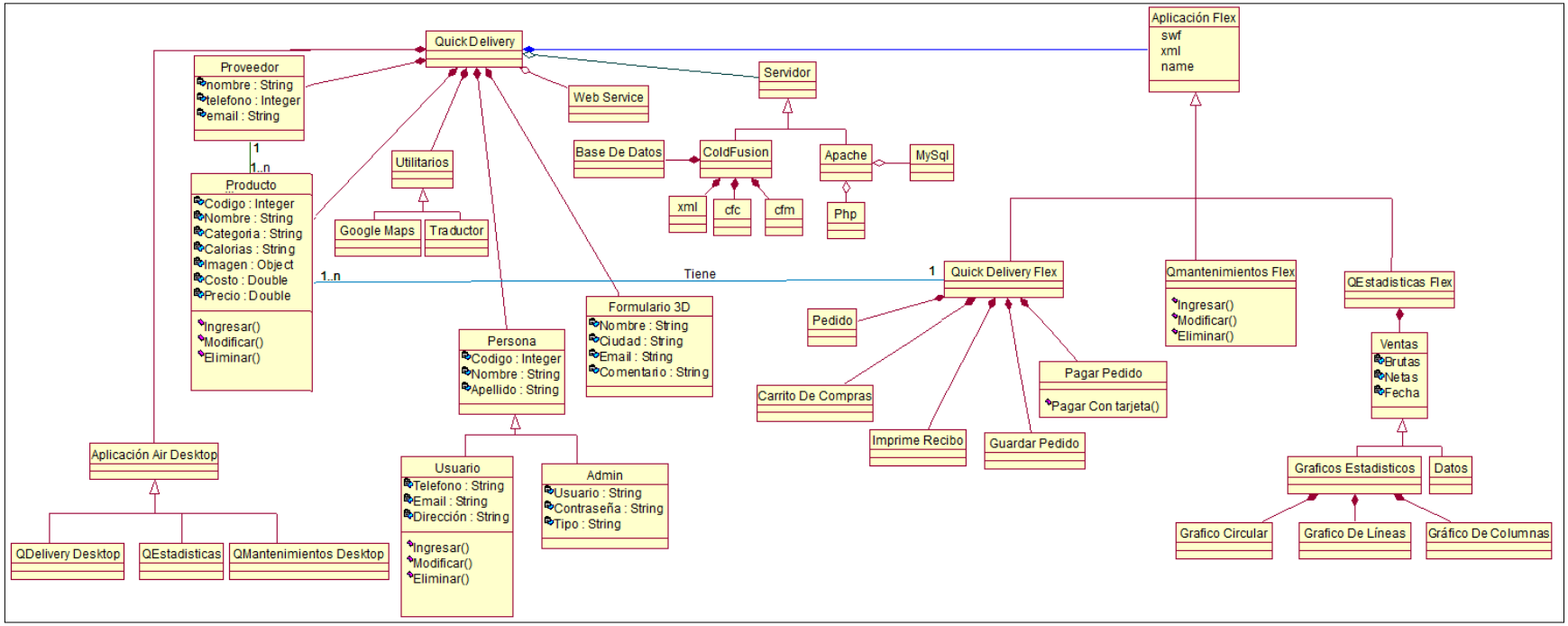


Figura 3. 6 Modelo De Contenido Diagrama De Clases

3.7.2. Modelo de interacción

El modelo de interacción detalla el motivo mediante el cual se diseña la interacción en la interfaz, por lo que indica de forma general los comportamientos del sistema frente a las distintas acciones que el usuario pueda observar.

3.7.2.1. Casos de uso

3.7.2.1.1. Caso de uso administrar sistema

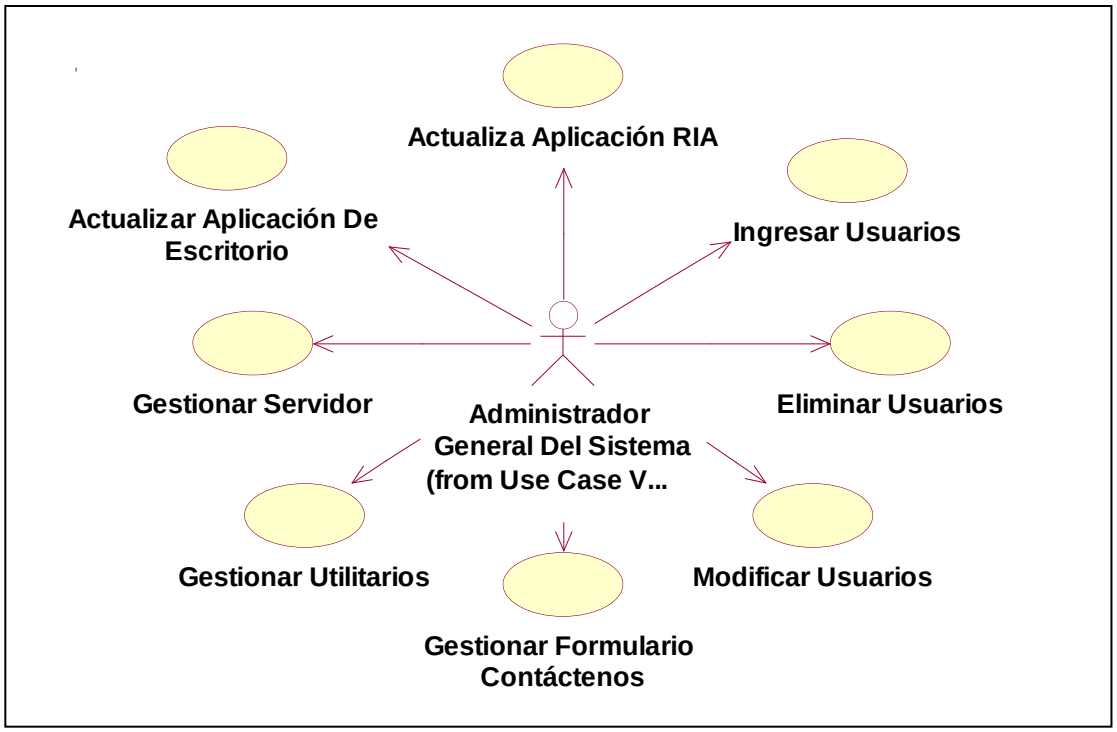


Figura 3.7 Caso De Uso Administración del sistema

Detalle caso de uso administrar sistema

Caso de uso 1	**Ingresar Usuarios**
Actor:	Administrador General Del Sistema
***Descripción*:**	El Administrador puede ingresar nuevos usuarios que sean de tipo administrador para ciertas áreas del sistema. Hay usuarios que se encargan de la gestión de ventas y otros que se encargan de los productos.
Flujo Normal:	1. Ingresa los datos del nuevo usuario. 2. Guarda el usuario.

Caso de uso 2	**Modificar Usuario**
Actor:	Administrador General Del Sistema
Descripción:	El Administrador puede modificar los usuarios existentes.
Flujo Normal:	1. Escoge el usuario a modificar. 2. Modifica los datos del usuario.

Caso de uso 3	**Eliminar Usuario**
Actor:	Administrador General Del Sistema
***Descripción*:**	El Administrador puede eliminar a un usuario existente.
Flujo Normal:	1. Escoge el usuario a eliminar. 2. Elimina el usuario.

Caso de uso 4	**Mantenimiento Aplicación RIA**
Actor:	Administrador General Del Sistema
***Descripción*:**	El administrador da mantenimiento a todo el sistema Quick Delivery.
Flujo Normal:	1. Administrador ingresa al sistema y actualiza y da mantenimiento a la aplicación RIA Quick Delivery.

Caso de uso 5	**Actualizar Aplicación De Escritorio**
Actor:	Administrador General Del Sistema
***Descripción*:**	El administrador actualiza las distintas aplicaciones de escritorio AIR.
Flujo Normal:	1. Administrador escoge la aplicación Air que se va actualizar y carga la nueva versión al sistema.

Caso de uso 6	**Gestionar Servidor**
Actor:	Administrador General Del Sistema
***Descripción*:**	El administrador gestiona los servidores ColdFusion y Apache, y los web services junto con la base de datos.
Flujo Normal:	1. El administrador ingresa a los servidores y da mantenimiento a la base de datos y web service.

Caso de uso 7	**Gestionar Formulario Contáctenos**
Actor:	Administrador General Del Sistema
***Descripción*:**	El administrador gestiona el formulario 3D Contáctenos para los comentarios y sugerencias de los clientes.
Flujo Normal:	1. El administrador ingresa al sistema. 2. Lee los comentarios y sugerencias de cada usuario.

Caso de uso 8	**Gestionar Utilitarios**
Actor:	Administrador General Del Sistema
***Descripción*:**	El administrador gestiona las aplicaciones APIS añadidas al sistema.
Flujo Normal:	1. El administrador ingresa al sistema. 2. Añadir una aplicación API.
Flujo Alternativo:	2. Eliminar una aplicación API.

3.7.2.1.2 Caso de uso Usuario Registrado

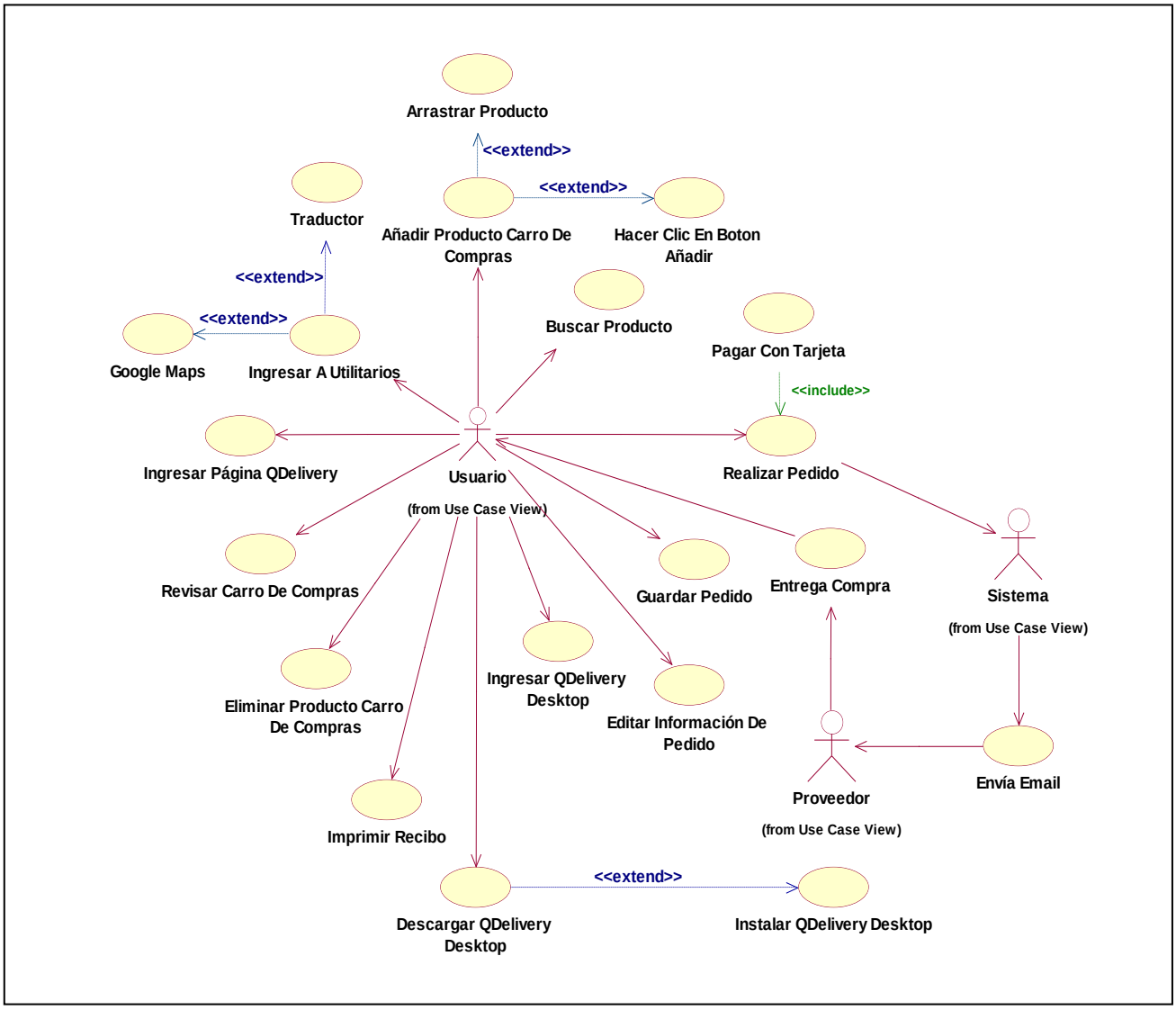


Figura 3. 8 Caso De Uso Usuario Registrado

Detalle de caso de uso registrado

Caso de uso 1	**Ingresar Pagina QDelivery**
Actor:	Usuario
***Descripción*:**	El usuario puede ingresar a la página sin necesidad de ser un usuario registrado
Flujo Normal:	1. Usuario ingresa a www.quickdelivery.com

Caso de uso 2	**Buscar Producto**
Actor:	Usuario
***Descripción*:**	El usuario puede buscar el producto a través de las distintas categorías que dispone el sistema Quick Delivery.
Flujo Normal:	1. Usuario escoge la categoría. 2. Busca el producto que desee de esa categoría.

Caso de uso 3	**Añadir Producto A Carro De Compras**
Actor:	Usuario
***Descripción*:**	El usuario puede ir escogiendo e ir añadiendo los productos al carro de compras.
Flujo Normal:	1. Usuario escoge la categoría. 2. Busca el producto que desee de esa categoría. 3. Añade el producto al carro de compras.

	Punto De Extensión (Arrastrar Producto): El usuario puede escoger y arrastrar el producto hacia el carro de compras. **Punto De Extensión (Hacer Clic En Botón Añadir):** EL usuario puede hacer click en el botón "Añadir" para agregar el producto.

Caso de uso 4	**Revisar Carro De Compras**
Actor:	Usuario
***Descripción*:**	El usuario puede revisar el contenido del carro de compras y el total de la compra.
Flujo Normal:	1. Usuario hace click en el botón "Ver Carrito". 2. Se muestra una pantalla con el detalle de los productos, su cantidad y valor total. 3. Usuario puede aumentar la cantidad del producto.

Caso de uso 5	**Eliminar Producto De Carro De Compras**
Actor:	Usuario
***Descripción*:**	El usuario puede eliminar el producto del carro de compras.
Flujo Normal:	1. Usuario ingresa en Carro De Compras. 2. Hace click en el botón "Eliminar" del producto que ya no desee. 3. Se resta el valor del producto eliminado.

Caso de uso 6	**Realizar Pedido**
Actor:	Usuario, Sistema
***Descripción*:**	El usuario realiza el pedido de la compra.
Flujo Normal:	1. Include (Pagar Tarjeta De Crédito) 2. Usuario hace click en el botón "Pedido" 3. Ingresa la información solicitada como: Nombre, Teléfono, Dirección, Email y se muestra la fecha actual. 4. Ingresa los datos de la Tarjeta De Crédito, escoge: Tarjeta, número de tarjeta, fecha de expiración. 5. Se detalla los datos del recibo. 6. Hace click en "Finalizar Compra". 7. Hace click en "Imprimir Recibo". 8. Pedido realizado.
Flujo Alternativo:	3. El sistema valida que los datos ingresados estén llenados correctamente, caso contrario muestra un mensaje de "Datos Llenados incorrectamente" y muestra los campo que hay que corregir. 4. El sistema verifica que los datos de la tarjeta sean correctos, caso contrario muestra un mensaje de "Datos Incorrectos". 6. Hace click en "Editar Información" y muestra nuevamente la pantalla con los datos personales para modificar.

Caso de uso 7	**Envía Email**
Actor:	Sistema, Proveedor
***Descripción*:**	El sistema se encarga de enviar un correo electrónico al proveedor con el detalle de la compra para la entrega del pedido.
Flujo Normal:	1. Sistema envía email con el detalle de la compra y los datos del cliente. 2. Proveedor recibe email con datos del pedido.

Caso de uso 8	**Entrega compra**
Actor:	Proveedor, Usuario
***Descripción*:**	El proveedor se encarga de entregar los productos de la compra en el domicilio del cliente.
Flujo Normal:	1. Proveedor entrega compra a usuario en la dirección registrada en la página Quick Delivery.

Caso de uso 9	**Guardar Carro De Compras**
Actor:	Usuario
***Descripción*:**	El usuario puede guardar su pedido para luego regresar a Quick Delivery y seguir con la compra.
Flujo Normal:	2. Usuario ingresa en Carro De Compras. 3. Se muestra todos los productos escogidos con su cantidad y valor total. 4. Hace click en "Guardar Para Luego". 5. Usuario sale del sistema de Quick Delivery. 6. Usuario ingresa al sistema Quick Delivery y encuentra los productos que guardo la última vez que ingresó en el carro de compras.

Caso de uso 10	**Imprimir recibo**
Actor:	Usuario
***Descripción*:**	El usuario puede imprimir su recibo.
Flujo Normal:	1. Usuario finaliza la compra. 2. Hace click en el botón "Imprimir Recibo".

Caso de uso 11	**Editar Información De Pedido**
Actor:	Usuario
***Descripción*:**	El usuario puede editar su información antes de terminar el pedido.
Flujo Normal:	1. Usuario hace click en "Editar información". 2. Se muestra la pantalla con la información ingresada anteriormente. 3. Se modifica los datos que el usuario requiera cambiar. 4. Continúa con la finalización del pedido.

Caso de uso 12	**Ingresar a utilitarios**
Actor:	Usuario
***Descripción*:**	El usuario puede ingresar a la opción de utilitarios en la aplicación.
Flujo Normal:	1. Usuario ingresa en utilitarios. 2. Escoge la aplicación. **Punto De Extensión (Ingresar en Google Maps)**: Ingresa en el mapa y puede hacer consultas de ubicación de lugares o hacer búsquedas. **Punto De Extensión (Ingresar en traductor)**: Ingresa en el traductor y tiene la opción de traducir palabras, textos en cualquier idioma.

Caso de uso 13	**Descargar QDelivery Desktop**
Actor:	Usuario
***Descripción*:**	El usuario puede descargar desde la página principal www.quickdelivery.com la aplicación de escritorio Air para instalarla en su computador.
Flujo Normal:	1. Usuario escoge en las opciones de "Aplicaciones De Escritorio" QDelivery Desktop. 2. Se descarga la aplicación con extensión *.air* y la guarda. **Punto De Extensión (Instalar QDelivery Desktop)**: Instala la aplicación Air en su computador.

Caso de uso 14	**Ingresar QDelivery Desktop**
Actor:	Usuario
***Descripción*:**	El usuario puede ingresar a la aplicación de escritorio instalada en su computador.
Flujo Normal:	1. Usuario ingresa en la aplicación QDelivery Desktop instalada en su computador. 2. Realiza las mismas actividades que se encuentran en la página Qdelivery en Internet.

3.7.2.1.3 Caso de uso mantenimiento productos

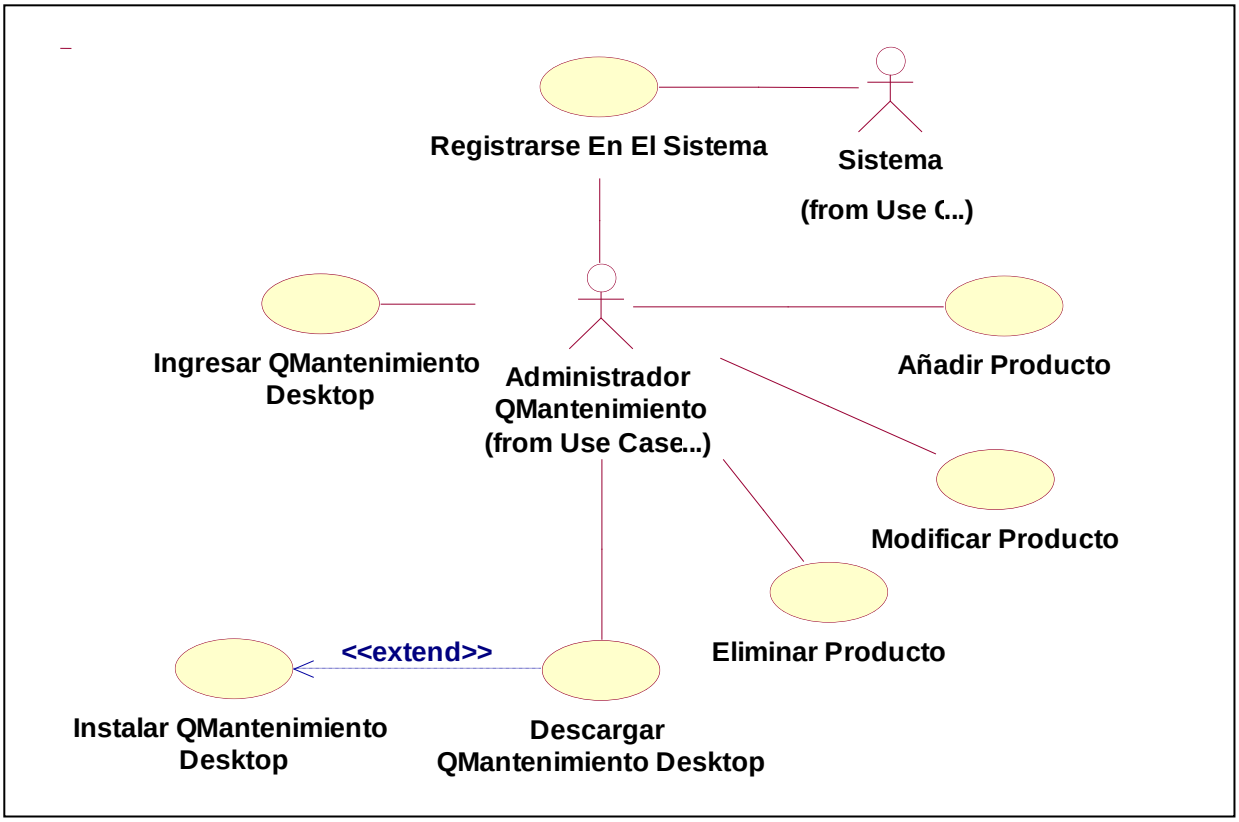


Figura 3.9 Caso De Uso Mantenimiento Productos

Detalle de caso de uso mantenimiento productos

Caso de uso 1	**Registrarse en el sistema**
Actor:	Administrador QMantenimiento, Sistema
***Descripción*:**	El administrador ingresa a la página principal donde debe registrarse para poder ingresar a la página.
Flujo Normal:	1. Administrador ingresa su usuario y contraseña. 2. Ingresa a página QMantenimiento.
Flujo Alternativo:	1. El sistema verifica que usuario y contraseña sean correctos, caso contrario muestra un mensaje "Usuario Incorrecto".

Caso de uso 2	**Añadir Producto**
Actor:	Administrador QMantenimiento
***Descripción*:**	El Administrador de QMantenimiento puede añadir productos a las categorías.
Flujo Normal:	1. Usuario escoge la categoría. 2. Ingresa los datos del nuevo producto. 3. Escoge una imagen para el producto. 4. Guarda el producto. 5. Se visualiza una ventana con el producto agregado.

Caso de uso 3	**Modificar Producto**
Actor:	Administrador QMantenimiento
***Descripción*:**	El usuario puede modificar los productos de las distintas categorías.
Flujo Normal:	1. Usuario escoge la categoría. 2. Busca el producto que desee de esa categoría. 3. Modifica los datos de ese producto. 4. Se visualiza una ventana con el producto modificado.

Caso de uso 4	**Eliminar Producto**
Actor:	Administrador QMantenimiento
***Descripción*:**	El usuario puede eliminar cualquier producto de las distintas categorías.
Flujo Normal:	1. Usuario escoge la categoría. 2. Busca el producto que desee eliminar de esa categoría. 3. Hace click en el botón de eliminar. 4. Se visualiza una ventana con el producto eliminado.

Caso de uso 5	**Eliminar Producto De Carro De Compras**
Actor:	Administrador QMantenimiento
***Descripción*:**	El usuario puede eliminar el producto del carro de compras.
Flujo Normal:	1. Usuario ingresa en Carro De Compras. 2. Hace click en el botón "Eliminar" del producto que ya no desee. 3. El total de la compra resta el valor del producto eliminado.

Caso de uso 6	**Descargar QMantenimiento Desktop**
Actor:	Administrador QMantenimiento
***Descripción*:**	El administrador puede descargar desde la página principal www.quickdelivery.com la aplicación de escritorio Air para instalarla en su computador.
Flujo Normal:	1. Usuario escoge en las opciones de "Aplicaciones De Escritorio" QMantenimiento Desktop. 2. Se descarga la aplicación con extensión *.air* y la guarda. **Punto De Extensión (Instalar QMantenimiento Desktop)**: Se instala la aplicación Air Desktop en el computador.

Caso de uso 7	**Ingresar QMantenimiento Desktop**
Actor:	Administrador QMantenimiento
***Descripción*:**	El administrador puede ingresar a la aplicación de escritorio instalada en su computador.
Flujo Normal:	1. Usuario ingresa en la aplicación QMantenimiento Desktop instalada en su computador. 2. Realiza las mismas actividades que se encuentran en la página QMantenimiento en Internet.

3.7.2.1.4 Caso de uso estadísticas de ventas

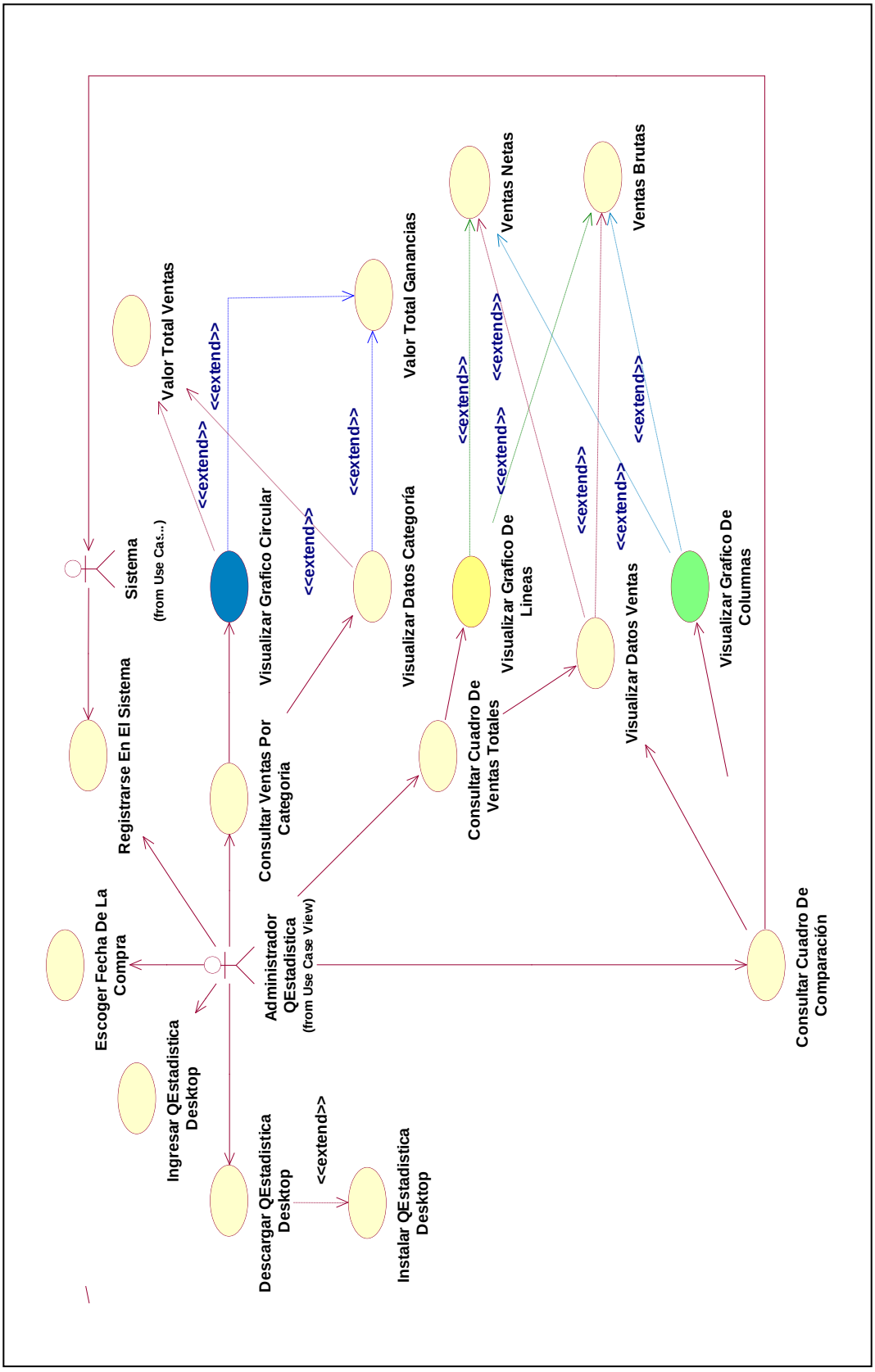


Figura 3.10 Caso De Uso Estadística Ventas

Detalle de casos de uso estadísticas de ventas

Caso de uso 1	**Registrarse en el sistema**
Actor:	Administrador QEstadistica, Sistema
***Descripción*:**	El administrador ingresa a la página principal donde debe registrarse para poder ingresar a la página.
Flujo Normal:	1. Administrador ingresa su usuario y contraseña. 2. Ingresa a página QEstadistica.

Caso de uso 2	**Escoger Fecha De La Compra**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador debe escoger un rango de fechas para visualizar las ventas realizadas.
Flujo Normal:	1. Administrador escoge fecha inicial y final para las ventas.

Caso de uso 3	**Consultar Ventas Por Categorías**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador escoge la categoría que desea visualizar, puede ver los datos o puede escoger el gráfico circular con las categorías e ir viendo el total de ganancias y el total de las ventas.
Flujo Normal:	1. Administrador escoge la categoría. 2. Escoge el gráfico circular y puede pasar el mouse sobre las categorías para que se vayan visualizando los datos de las ventas totales y ganancias. 3. Escoge los datos, donde se muestra una tabla con las categorías, ventas totales y ganancias.
Flujo Alternativo:	2. EL administrador puede escoger las ventas totales o las ganancias.

Caso de uso 4	**Visualizar Gráfico Circular**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede escoger el gráfico circular para visualizar las ventas y ganancias totales.
Flujo Normal:	1. Administrador escoge el cuadro circular. **Punto De Extensión (Valor Total Ventas)**: Visualiza los datos de las ventas totales. **Punto De Extensión (Valor Total Ganancias)**: Visualiza los datos de las ganancias totales.

Caso de uso 5	**Visualizar Datos Categoría**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede visualizar los datos de las ventas y ganancias en un cuadro de datos.
Flujo Normal:	1. Administrador escoge visualizar los datos. **Punto De Extensión (Valor Total Ventas)**: Visualiza los datos de las ventas totales. **Punto De Extensión (Valor Total Ganancias)**: Visualiza los datos de las ganancias totales.

Caso de uso 6	**Visualizar Gráfico De Líneas**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede escoger el gráfico circular para visualizar las ventas netas y ventas brutas.
Flujo Normal:	1. Administrador escoger el gráfico de líneas. **Punto De Extensión (Ventas Netas)**: Visualiza las ventas netas. **Punto De Extensión (Ventas Brutas)**: Visualiza las ventas brutas.

Caso de uso 7	**Visualizar Datos Ventas**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede visualizar los datos de las ventas netas y brutas.
Flujo Normal:	1. Administrador escoge visualizar los datos de las ventas. **Punto De Extensión (Ventas Netas)**: Visualiza los datos de las ventas netas. **Punto De Extensión (Ventas Brutas)**: Visualiza los datos de las ventas brutas.

Caso de uso 8	**Visualizar Gráfico De Columnas**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede escoger el gráfico de columnas para visualizar las ventas netas y ventas brutas.
Flujo Normal:	1. Administrador escoger el gráfico de columnas. **Punto De Extensión (Ventas Netas)**: Visualiza las ventas netas. **Punto De Extensión (Ventas Brutas)**: Visualiza las ventas brutas.

Caso de uso 9	**Consultar Cuadro De Ventas Totales**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede visualizar las ventas totales en el rango de fechas escogidas o puede escoger un cuadro con los datos de las ventas.
Flujo Normal:	2. Administrador escoge el cuadro de ventas totales. 3. Escoge el cuadro de líneas. 4. Escoge ventas brutas o ventas netas. 5. Se visualiza las fechas y el cuadro de líneas. 6. Escoge los datos y se muestra una tabla con las fechas, ventas brutas y ventas totales de todas las categorías.

Caso de uso 10	**Consultar Cuadro De Comparación**
Actor:	Administrador QEstadistica
***Descripción*:**	El administrador puede visualizar las ventas brutas y netas en un cuadro de comparación y un cuadro de barras.
Flujo Normal:	4. Administrador escoge el cuadro de comparación. 5. Escoge el cuadro de barras. 6. Se muestra el cuadro con las fechas y las ventas netas y brutas en comparación. 7. Escoge los datos y se muestra una tabla con las fechas, ventas brutas y ventas totales de todas las categorías.

Caso de uso 11	**Descargar QDelivery Desktop**
Actor:	Administrador QEstadistica
***Descripción*:**	El usuario puede descargar desde la página principal www.quickdelivery.com la aplicación de escritorio Air para instalarla en su computador.
Flujo Normal:	1. Usuario escoge en las opciones de "Aplicaciones De Escritorio" QEstadistica Desktop. 2. Se descarga la aplicación con extensión *.air* y la guarda. **Punto De Extensión (Instalar QEstadistica Desktop)**: Se instala la aplicación Air Desktop en el computador.

Caso de uso 12	**Ingresar QDelivery Desktop**
Actor:	Administrador QEstadistica
***Descripción*:**	El usuario puede ingresar a la aplicación de escritorio instalada en su computador.
Flujo Normal:	1. Usuario ingresa en la aplicación QEstadistica Desktop instalada en su computador. 2. Realiza las mismas actividades que se encuentran en la página Estadística en Internet.

3.7.2.2. Diagramas De Secuencia

Los diagramas de secuencia nos muestran la interacción entre objetos y el orden secuencial en el que van sucediendo dichas interacciones, es decir cómo se comunican los objetos entre sí.

Diagrama de secuencias para el funcionamiento del sistema Quick Delivery

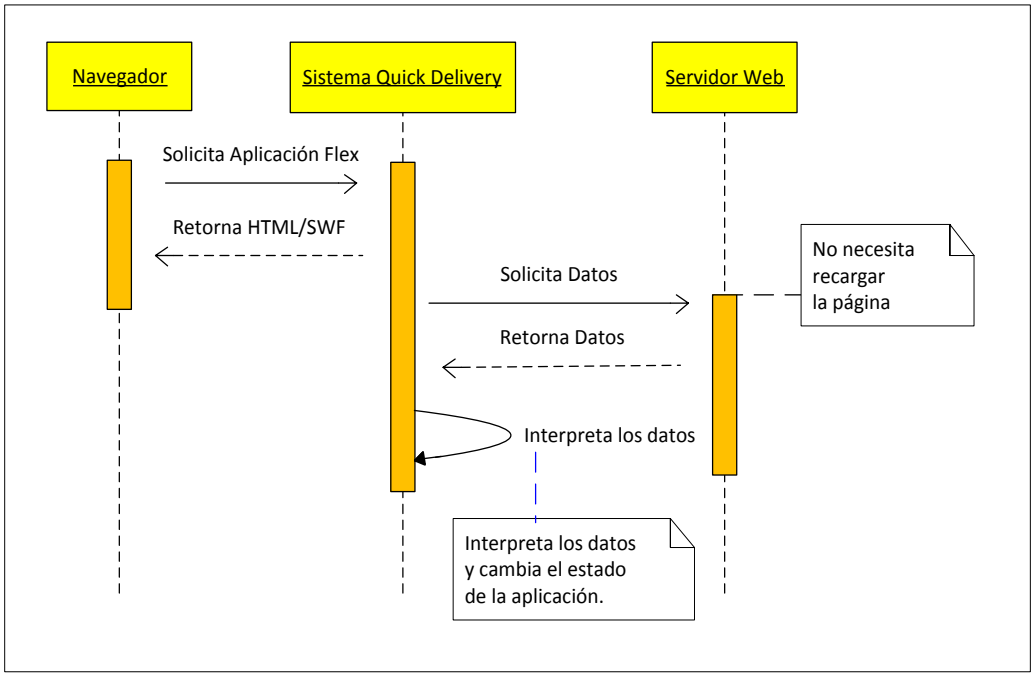


Figura 3.11 Diagrama De Secuencias Para Quick Delivery

Diagrama de secuencias para realizar compras

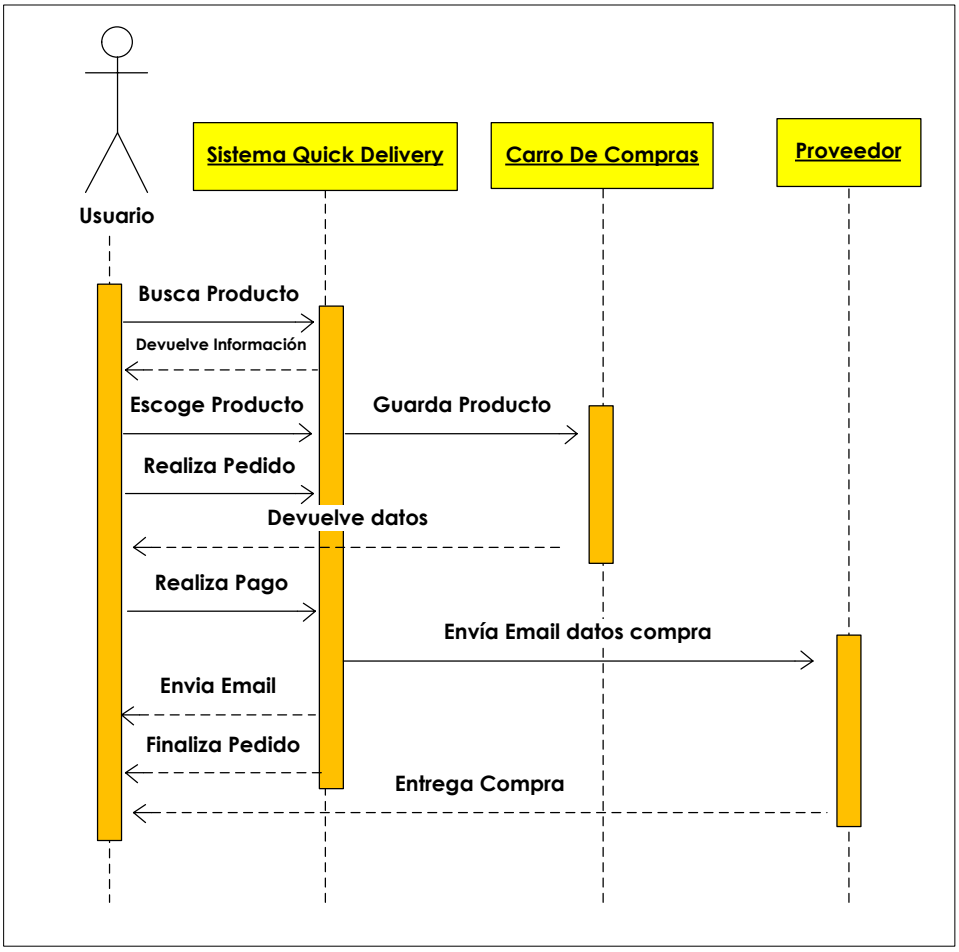


Figura 3.12 Diagrama De Secuencias Para Realizar Compra

Diagrama de secuencias para guardar carro de compras para luego

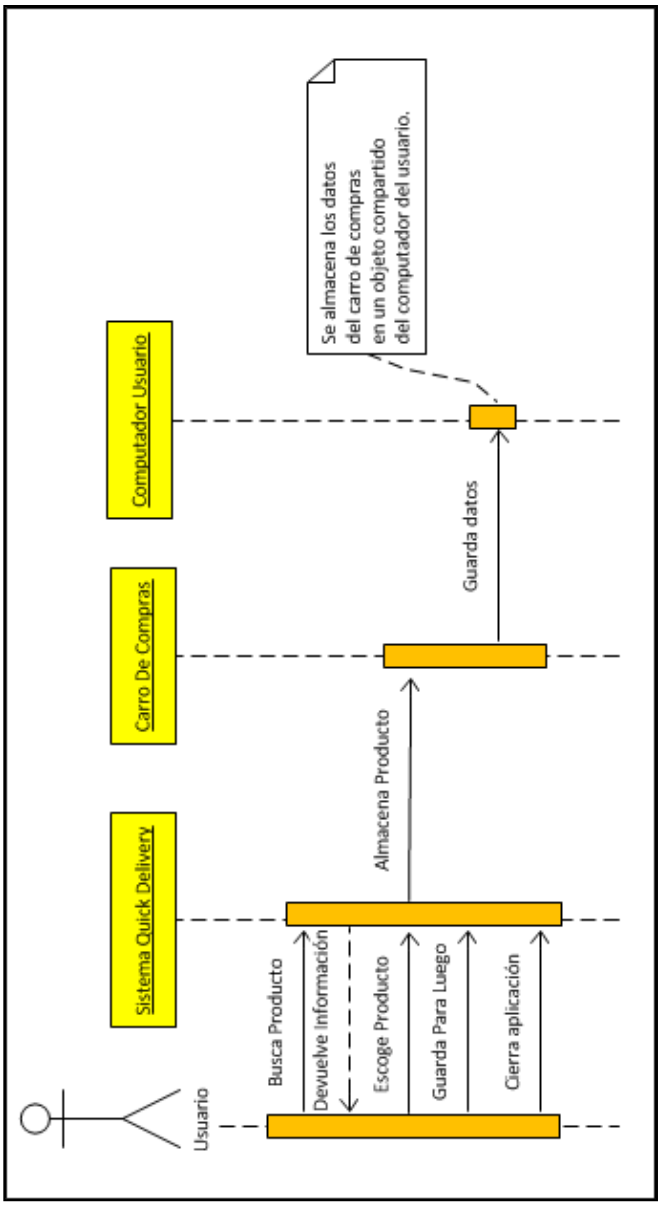


Figura 3.13 Diagrama De Secuencias Para Guardar Carro De Compras Para Luego

Diagrama de secuencias para recupera datos guardados en carro de compras, después de haber guardado para luego:

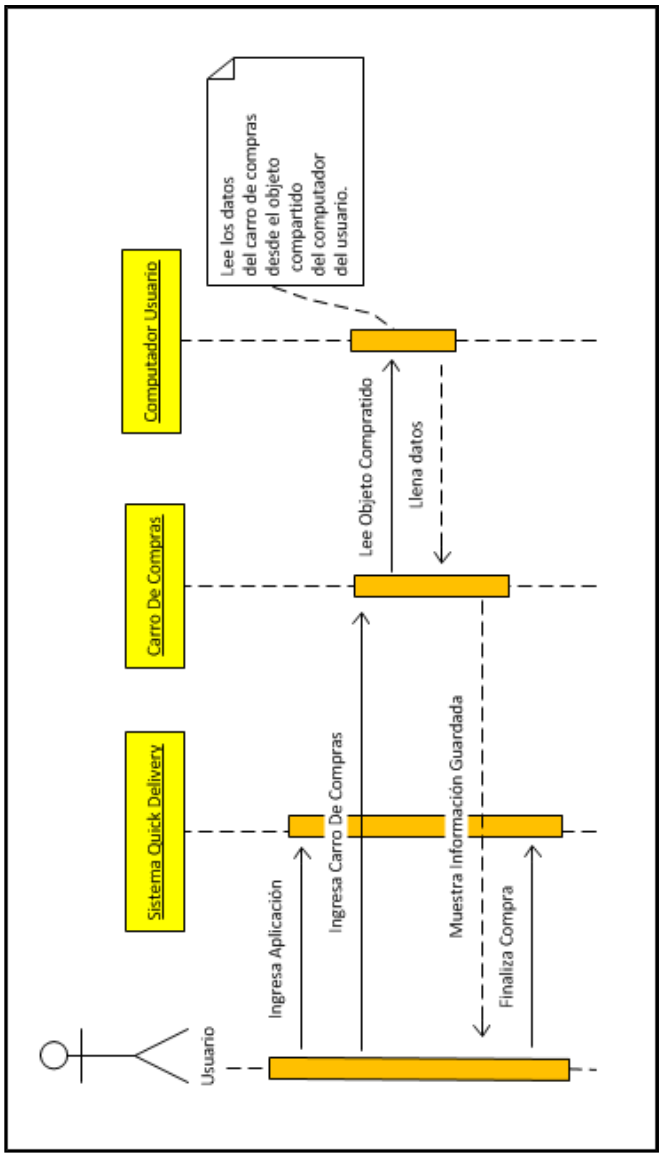


Figura 3.14 Diagrama De Secuencias Para Recuperar Datos De Carro De Compras

3.7.2.3. Diagrama De Estados

Diagrama de estados para descargar las aplicaciones Air Desktop:

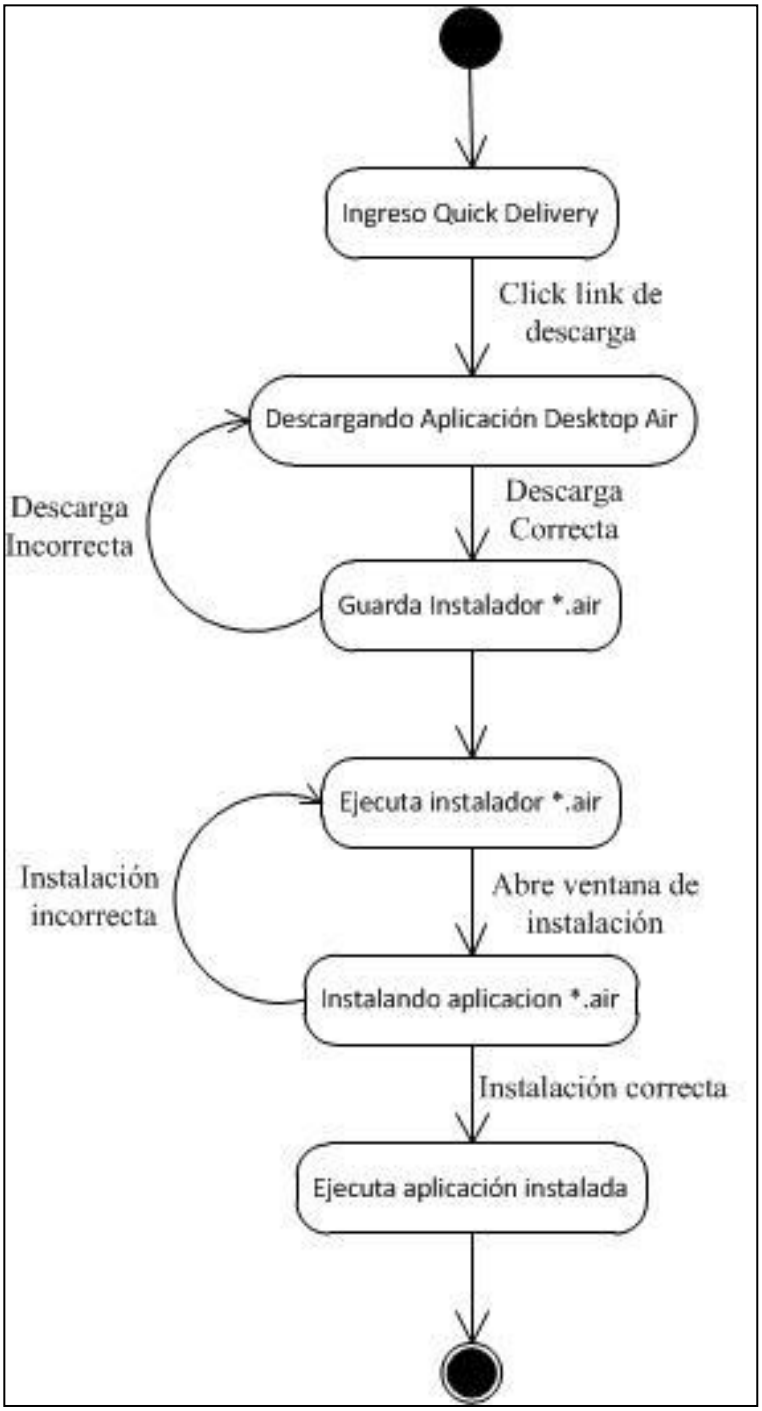


Figura 3.15 Diagrama De Estado Para Descargar Aplicación Air Desktop

Diagrama de estados para pedido:

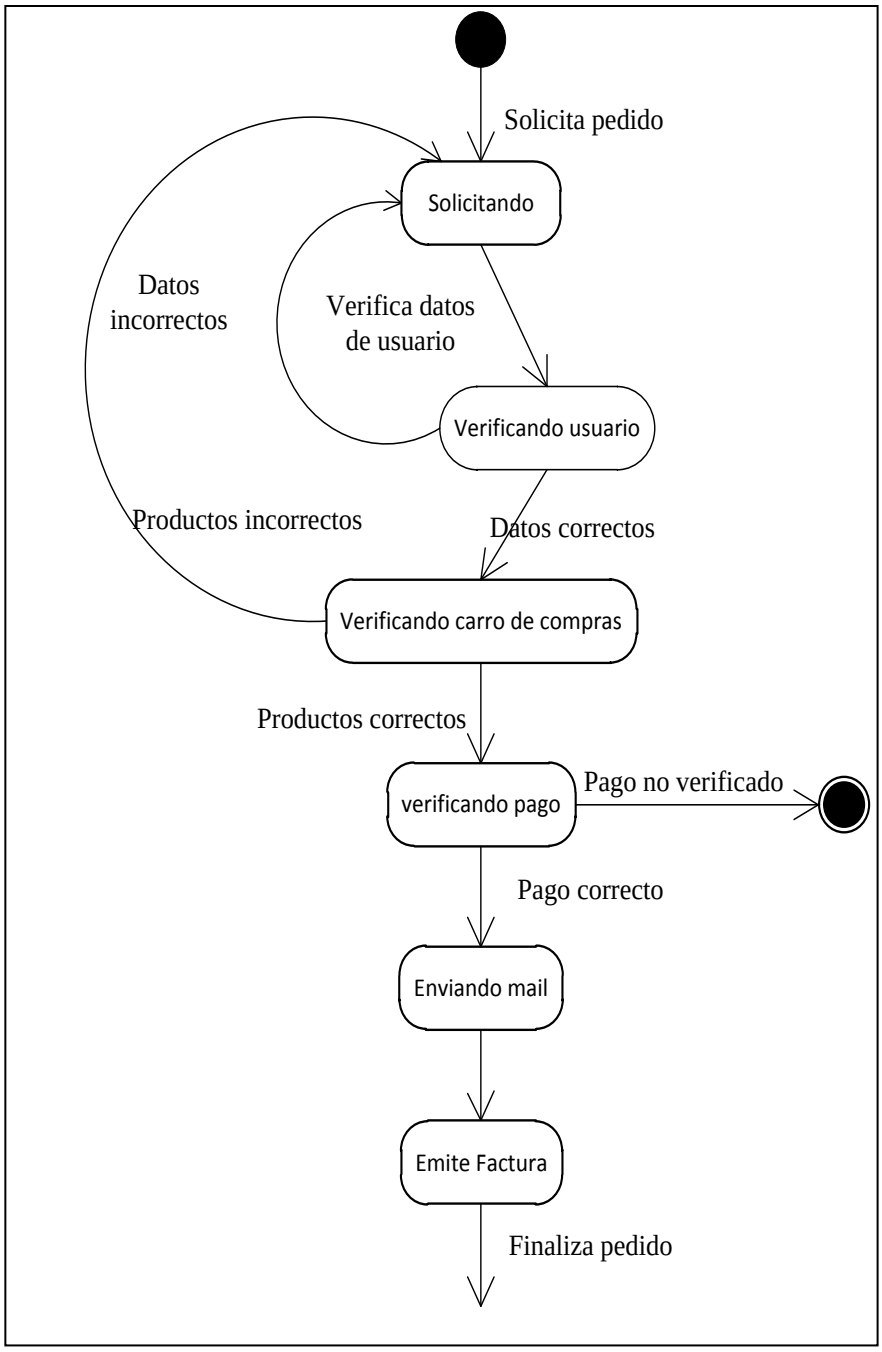


Figura 3.16 Diagrama De Estado Para Realizar Pedido

Diagrama de estados para carro de compras:

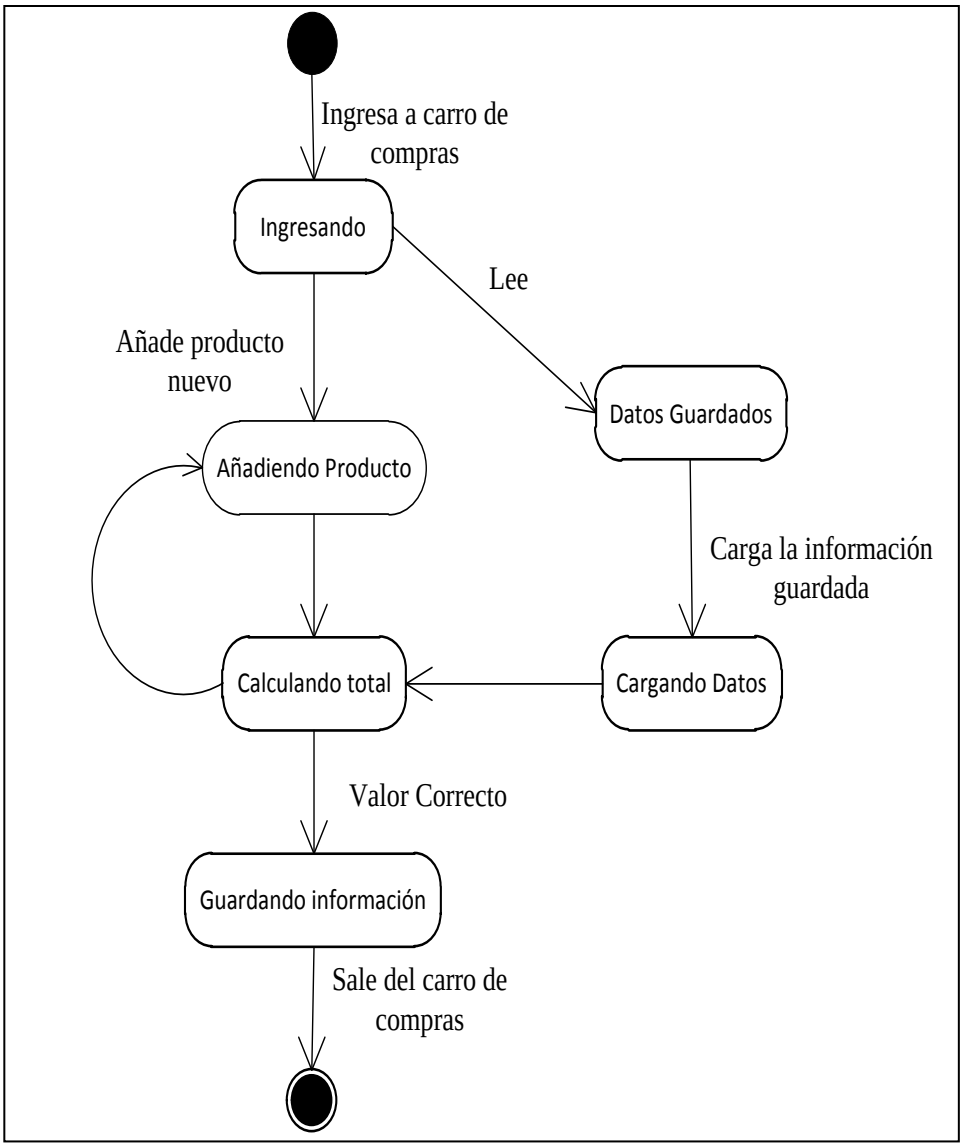


Figura 3.17 Diagrama De Estado Para Carro De Compras

3.7.2.4. Prototipo De Interfaz De Usuario

Es necesario realizar un prototipo del diseño con anticipación para así obtener requerimientos adicionales que se necesiten y así en el diseño del sistema obtener un diseño exacto de lo que se necesita y se va a realizar con la aplicación.

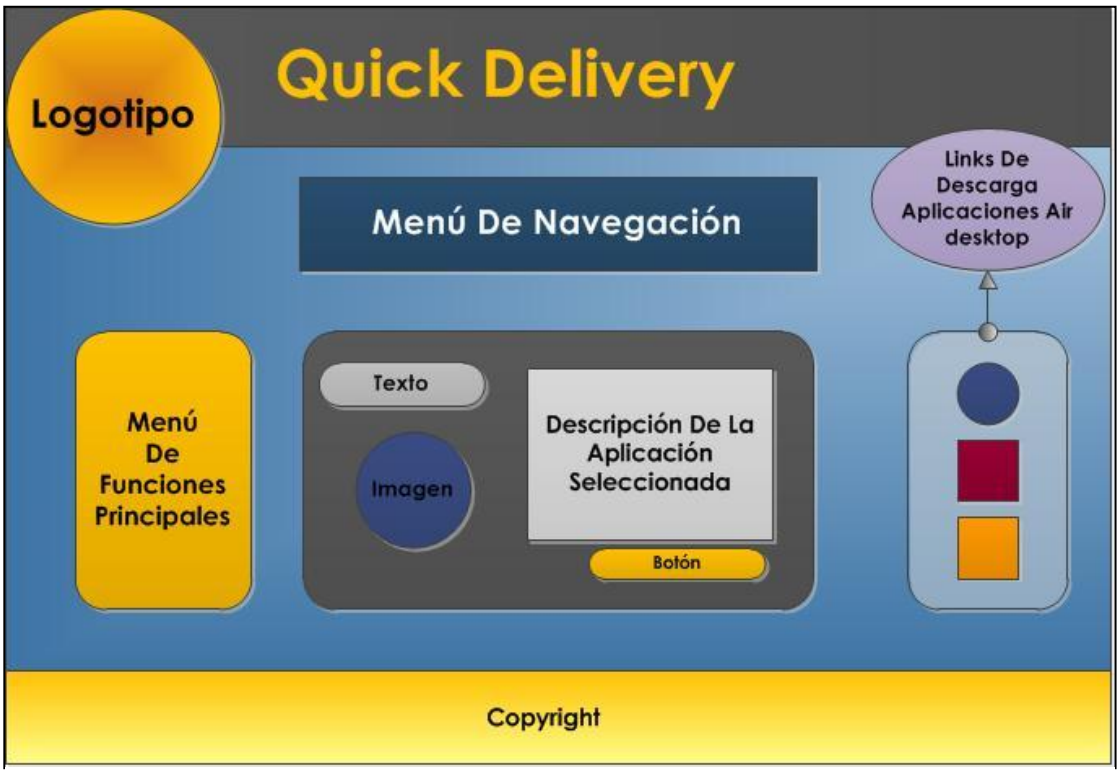


Figura 3. 18 Prototipo De Interfaz De Usuario

3.7.3. Modelo funcional

Se modelarán los aspectos dinámicos del sistema y los procesos computacionales, y a su vez visualizar, especificar, construir y documentar la dinámica de una sociedad de objetos.

3.7.3.1. Diagrama De Actividad

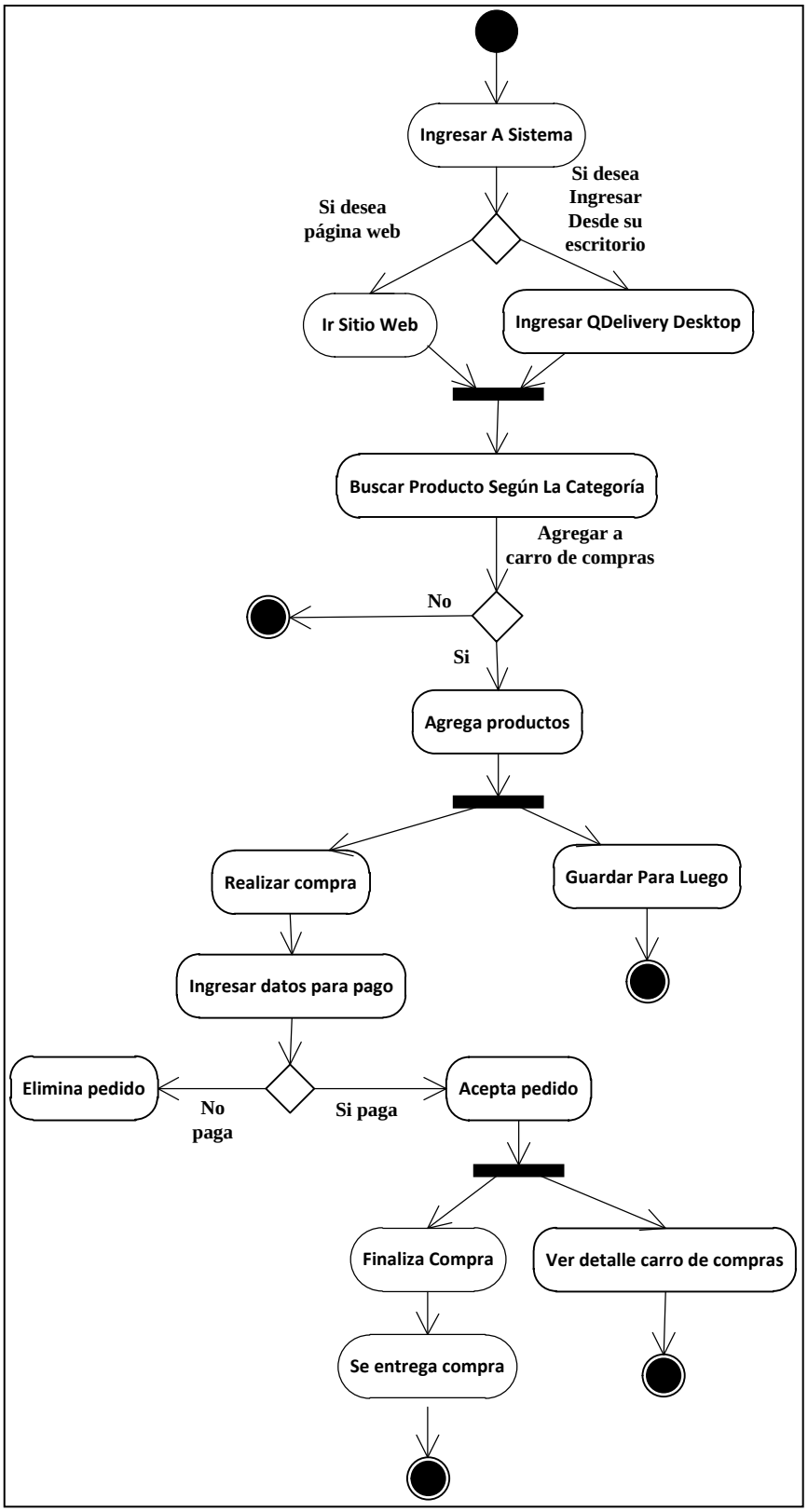


Figura 3.19 Diagrama De Actividad Compra

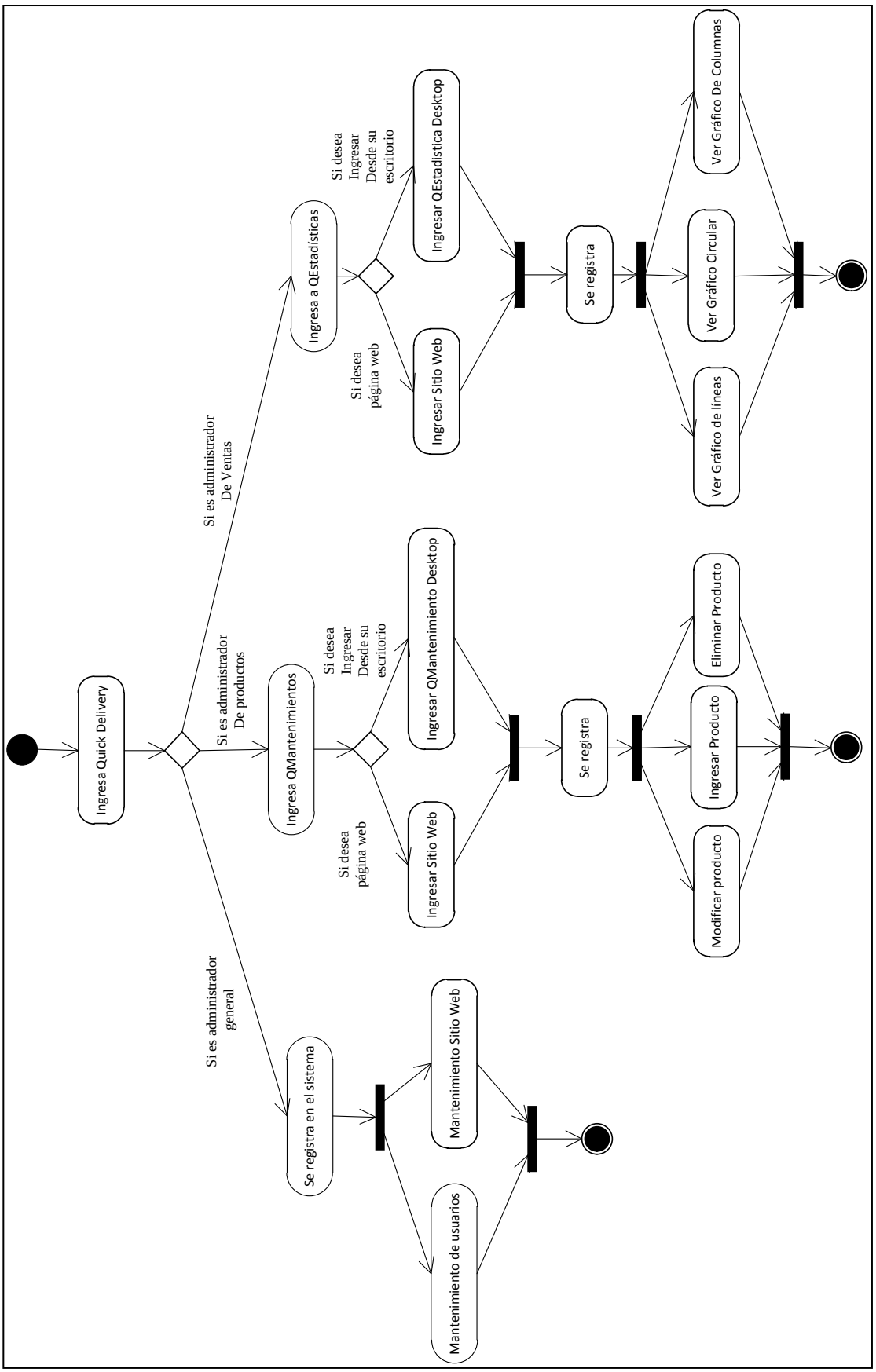


Figura 3.20 Diagrama De Actividad Sistema Quick Delivery

3.7.4. Modelo funcional

Describiremos el entorno y la infraestructura de la WebApp estableciendo los componentes web, que nos permiten indicar que patrones de arquitectura se desean aplicar y cómo los usuarios interactúan con el sistema.

3.7.4.1. Diagrama De Despliegue

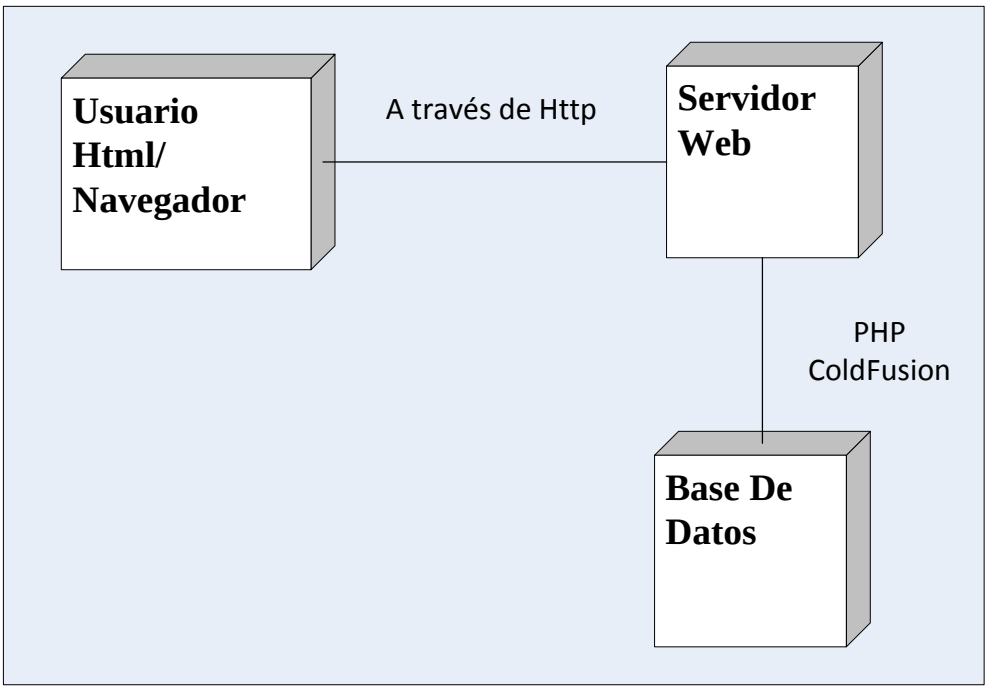


Figura 3.21 Diagrama De Despliegue Sistema Quick Delivery

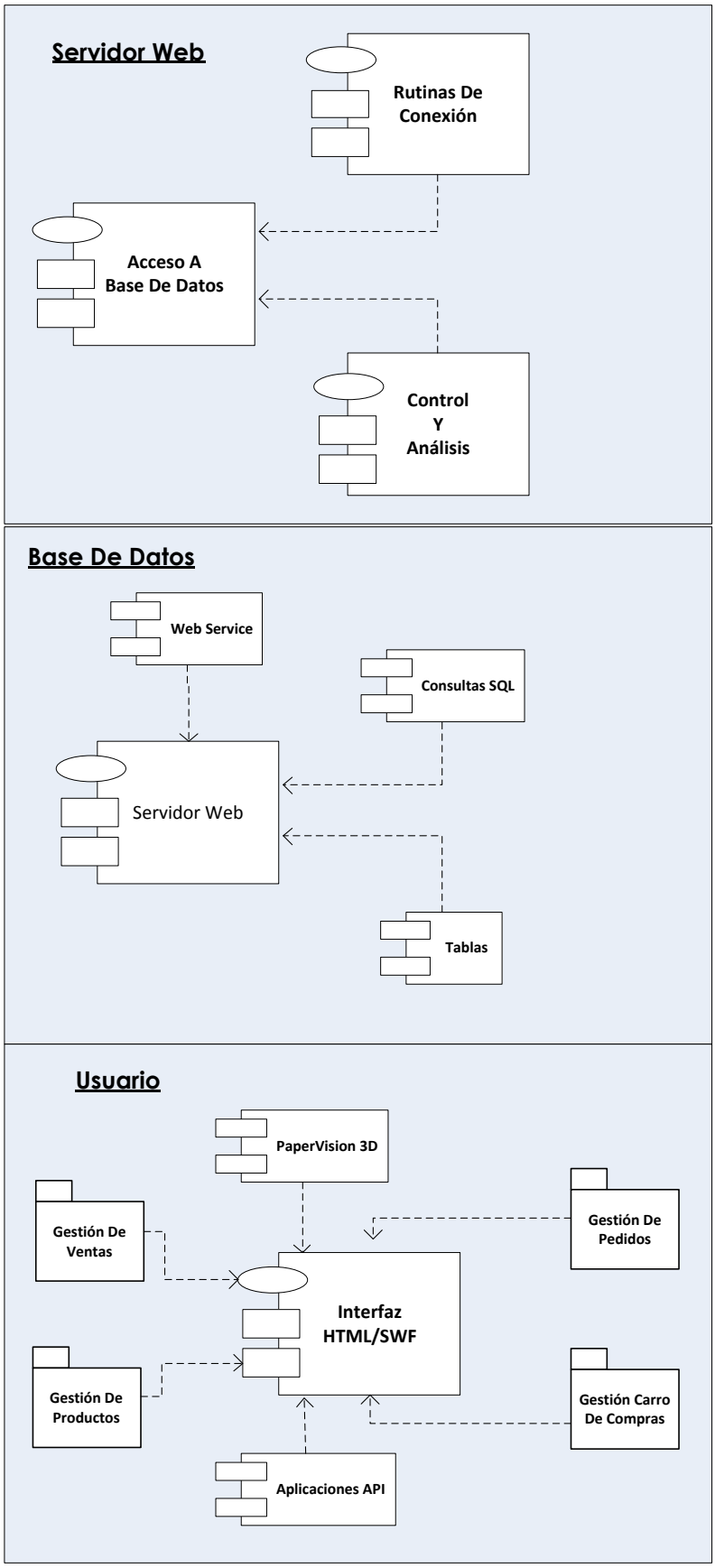


Figura 3.22 Diagrama De Componentes Sistema Quick Delivery

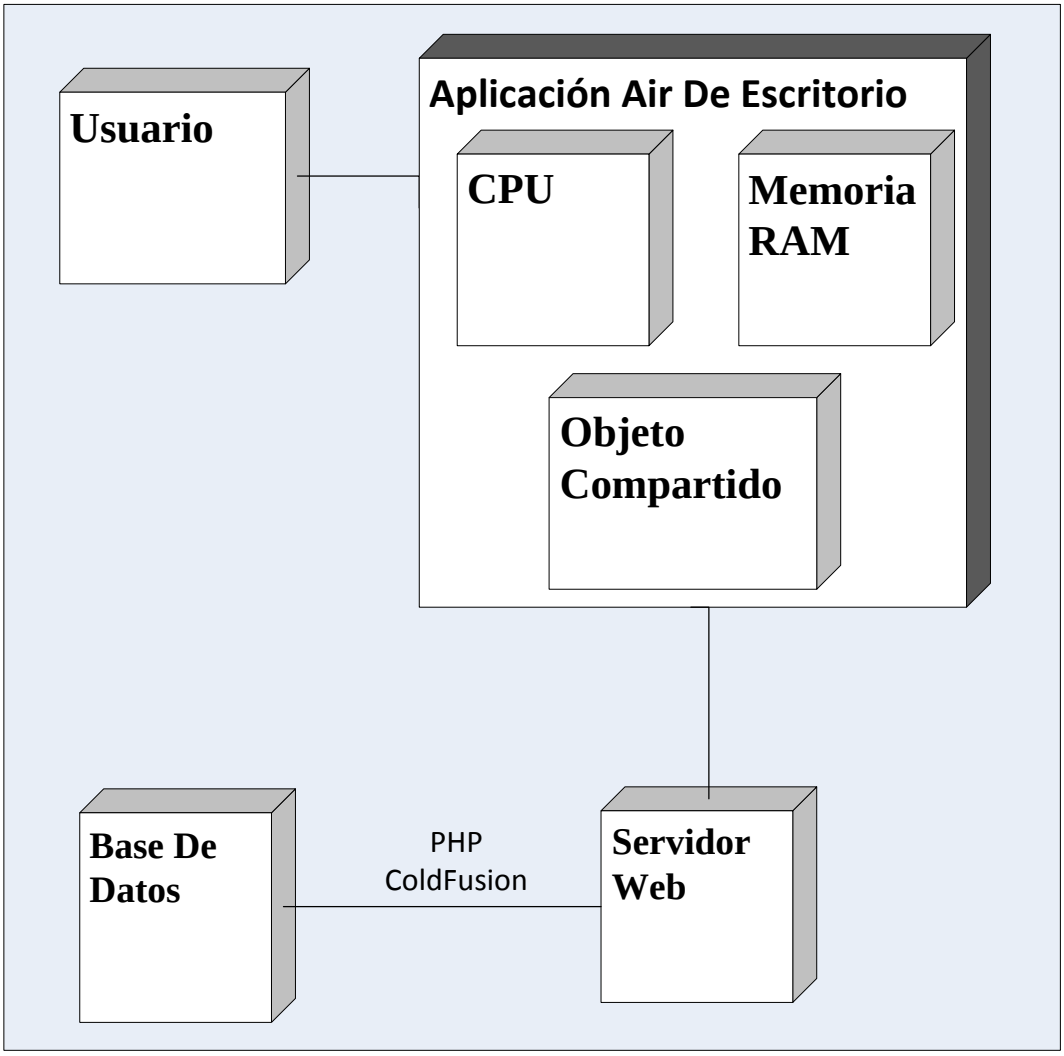


Figura 3.23 Diagrama De Despliegue Aplicación Air Desktop

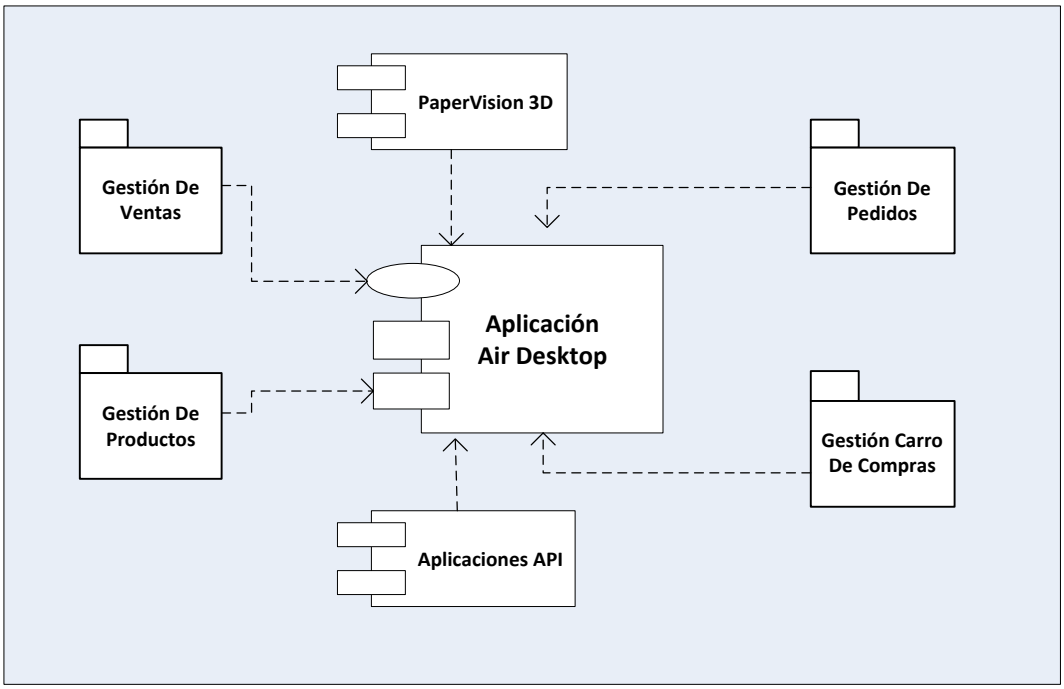


Figura 3.24 Diagrama De Componentes Aplicación Air Desktop

3.7.5. Análisis Relación-Navegación

El análisis de relación-navegación es la descripción de elementos y herramientas de navegación, analizando su función orientadora.

Los elementos de navegación y orientación tienen como función básica informar constantemente al usuario acerca de dónde se encuentra, dónde ha estado y hacia dónde puede ir. El objetivo es no perder al usuario.

- **No perder al usuario:** Las dos formas básicas para no perder al usuario son que no se aburra y que no se pierda navegando. Los contenidos o servicios de nuestra web son del interés del usuario por lo que se propone varias formas para que esta encuentre lo que está buscando.

- **Coherencia del diseño**

  La primero es que se indica a nuestro usuario que sigue estando en nuestra web manteniendo una coherencia en el diseño, es decir, una uniformidad en la estructura de las páginas que forman nuestro sitio, y también en lo colores empleados.

  La cabecera que mostramos en el home page está en todas las páginas del sitio y esta no desaparece. Por ejemplo, se disminuirá en ciertas ocasiones su tamaño para no desperdiciar demasiado espacio visual. La utilización de los colores se mantiene con cierta uniformidad.

- **Jerarquía Visual**

  Los usuarios (en occidente) leen de izquierda a derecha y de arriba hacia abajo. Mantendremos la jerarquía visual y así podemos indicarle al usuario constantemente dónde está. El elemento que se coloca en la esquina superior izquierda es el logo de nuestra página de "Quick Delivery", este es el elemento de mayor nivel jerárquico, con lo que le indicaremos que el resto de elementos son subelementos de éste. Es decir, que todos serán parte del logo de nuestra web.

3.7.6. Modelo de datos

Es importante la realización de un modelo de datos para describir la estructura de la base de datos de nuestro sistema, para esto se ha realizado un Modelo Entidad-Relación y un Modelo Relacional que incluyen entidades, atributos y relaciones.

3.7.6.1. Modelo Entidad-Relación

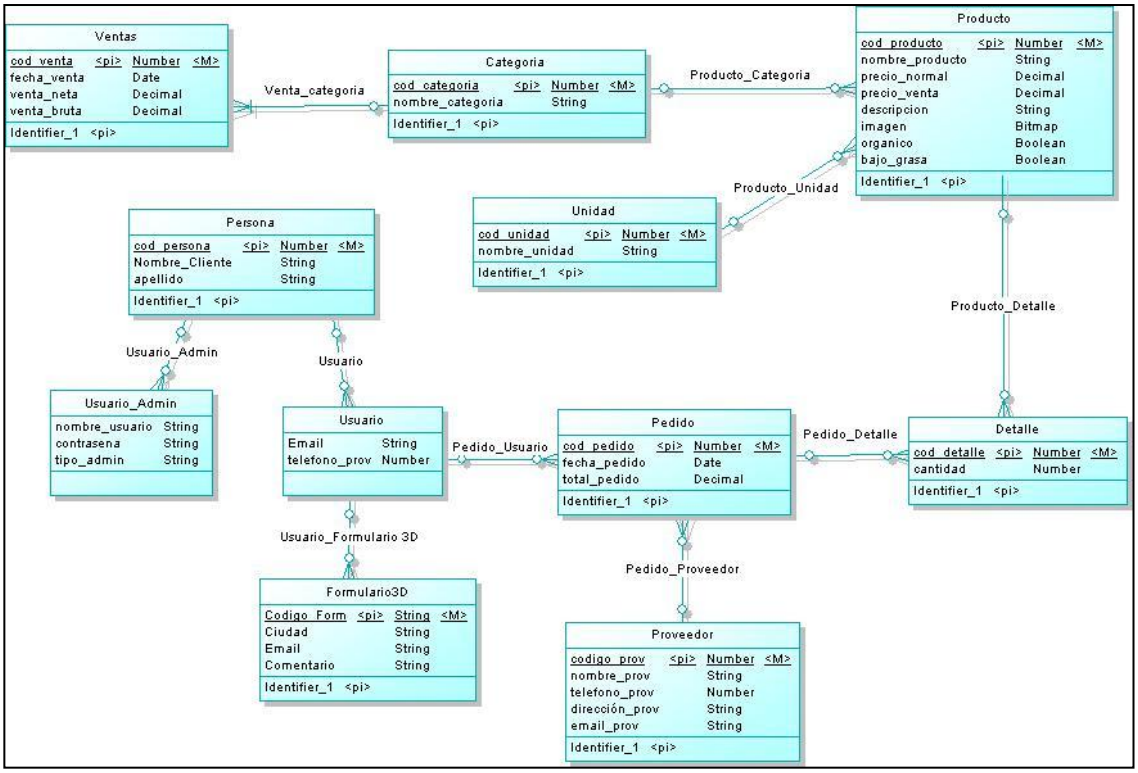


Figura 3.25 Modelo Entidad-Relación Quick Delivery

3.7.6.2. Modelo relacional

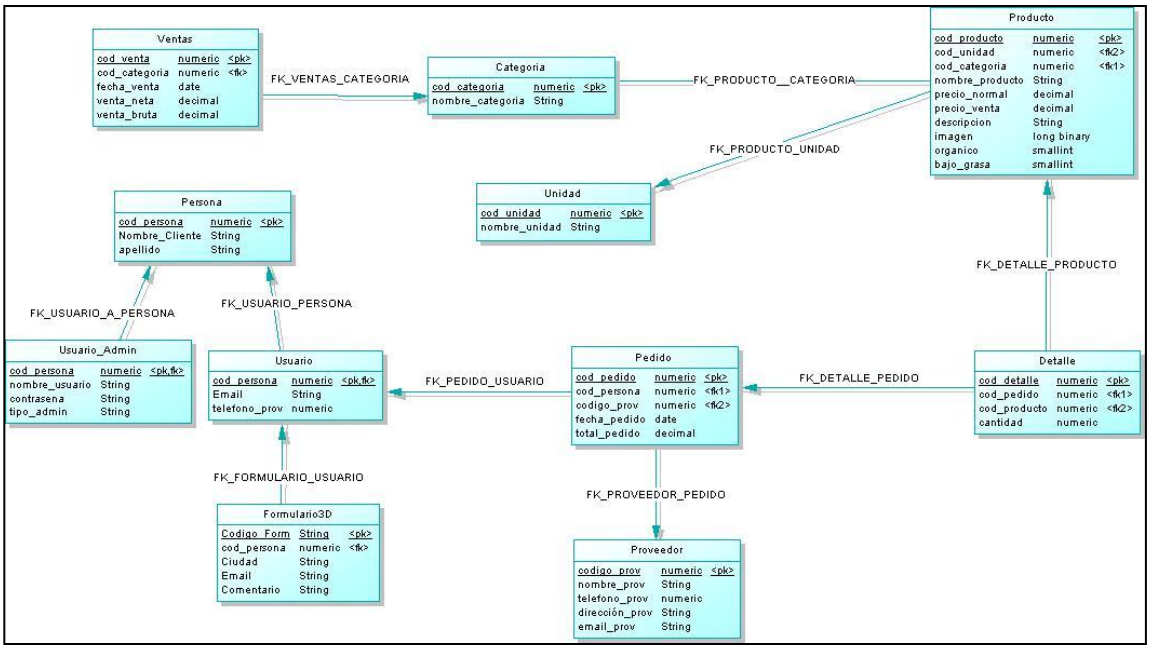


Figura 3.26 Modelo Relacional Quick Delivery

3.8. Conclusión

Gracias al análisis del sistema que pudimos realizar en base de la necesidades y requisitos que hemos obtenido tanto de los usuarios como de experiencias pasadas en el desarrollo de otros proyectos, hemos realizado los diagramas que nos construyen una visión clara del sistema que vamos a crear, entre los modelos que realizamos se incluyen casos de uso con el cual podemos saber cómo interactúan los actores con el sistema, también tenemos el modelo de Base de datos con el que definimos como se estructura los registros y la base de datos que usaremos para albergar la información, también tenemos los diagramas de secuencias y el diagrama de interactividad con los cuales podemos reconocer cual será el orden secuencial de las actividades y como interactuaran los componentes con los diferentes usuarios.

CAPÍTULO IV

CAPÍTULO 4

DISEÑO DEL SISTEMA QUICK DELIVERY

4.1 Introducción

En este capítulo realizaremos el diseño de nuestra página web con el propósito de recolectar todos los detalles que nos facilitará la creación de los componentes visuales de nuestra WebApp. Se implementarán todos los requisitos explícitos que se encuentran contemplados en el modelo de análisis realizados en el capítulo 3 y aquellos requisitos necesarios para los distintos tipos de usuarios que interactuarán en el sitio web.

Lo que procuraremos es diseñar a través de una estructura jerárquica los componentes que los usuarios utilizarán para que estos obtengan mejores beneficios al momento de navegar en nuestro sistema.

La importancia del diseño del software radica en la calidad, permitiendo describir todos los aspectos esenciales del sistema que estamos por desarrollar, y de esta manera generar resultados correctos para los usuarios.

Otro aspecto importante a tratar en este capítulo son el diseño de las interfaces, las cuales permiten la comunicación entre los usuarios y el sistema, permitiendo que los objetos sean atractivos y que los usuarios puedan utilizar el sitio web de la forma más intuitiva y amigable posible.

Además para brindar mayor estabilidad en nuestro sistema nos basaremos en la arquitectura MVC, para mejorar la arquitectura y la funcionalidad del sistema que estaremos desarrollando posteriormente.

4.2 Diseño de la interfaz de la Web App

El objetivo del diseño de la interfaz es que las aplicaciones y los objetos del sistema sean más atractivos y que la interacción con el usuario sea lo más intuitiva y amigable posible. El sistema está hecho para distintos usuarios, lo cual requiere que el mismo tenga una interfaz válida para todos los usuarios y todas las tareas que pueda realizar cada uno de estos. A continuación analizaremos las distintas interfaces según el modelo de análisis.

4.2.1. Refinamiento De La Información según el modelo de análisis.

Se proporcionará la información para todos los usuarios en un entorno visual sencillo que permita la fácil comunicación con el sistema y a su vez incluir objetos estándares en toda la aplicación, para que así los usuarios tengan el completo control sobre las funcionalidades del sitio. Además se incluirá links para la descarga de las aplicaciones Air De escritorio, las cuáles son exactamente iguales a las que se encuentran en el servidor y puedan ser instaladas fácilmente en cualquier computador.

- **Usuario Administrador:** Dentro de esta categoría se encuentran 3 tipos de usuarios administrativos, debido a que existen distintas áreas dentro del sistema que son administradas indistintamente por cada usuario, los cuales tendrán un usuario y contraseña para el ingreso en el sistema según sea su área.

  El administrador Master del sistema, es quién lleva la administración total del sistema, pudiendo añadir, modificar y eliminar usuarios; también realizará el mantenimiento general de la página y algo muy importante es la realización de los ejecutables para las aplicaciones Air de escritorio que son iguales a la página web que se encuentra en el servidor .

El administrador de productos será el encargado de añadir, modificar y eliminar los productos. Como la cantidad de información sobre los productos a venderse es grande, se ha establecido una interfaz sencilla y fácil de manejar, teniendo acceso en forma inmediata al contenido que desea e interactúe en forma sencilla con el sistema.

EL administrador de ventas es el encargado de visualizar los datos de las ventas, para esto se ha clasificado la información por fechas que se podrán escoger desde dos calendarios para escoger la fecha de inicio y fecha final que quiera consultar, también podrá escoger las distintas categorías de los productos desde un menú desplegable. Dicha información será visualizada en cuadros estadísticos de columnas, circular y de líneas, que se mostraran todos a la vez para que el usuario pueda trabajar con toda la información sin tener que cambiar el gráfico. Los componentes gráficos de la interfaz serán claros y de fácil identificación, teniendo fácil acceso a los contenidos de información.

- **Usuario Comprador:** Este usuario es quién realizará la compra de los productos y es quién navegará por casi la mayor parte de la aplicación.
  Para la visualización de los productos se realizará un menú que permite un rápido y cómodo acceso a los mismos, donde también se mostrará el detalle de cada producto. Para el usuario será fácil escoger el producto, ya que podrá desde hacer clic en un botón para agregar el producto hasta arrastrar y soltar el producto en el carro de compras.
  En el carro de compras se dispondrá del detalle de los productos así como también podrá guardar el pedido, para

que en otra sesión pueda seguir comprando sin que tenga que perder los datos del pedido anterior así este no haya culminado la compra; y para finalizar la compra podrá llenar sus datos en un formulario y culminar la misma.

4.3. Bosquejos de interfaces

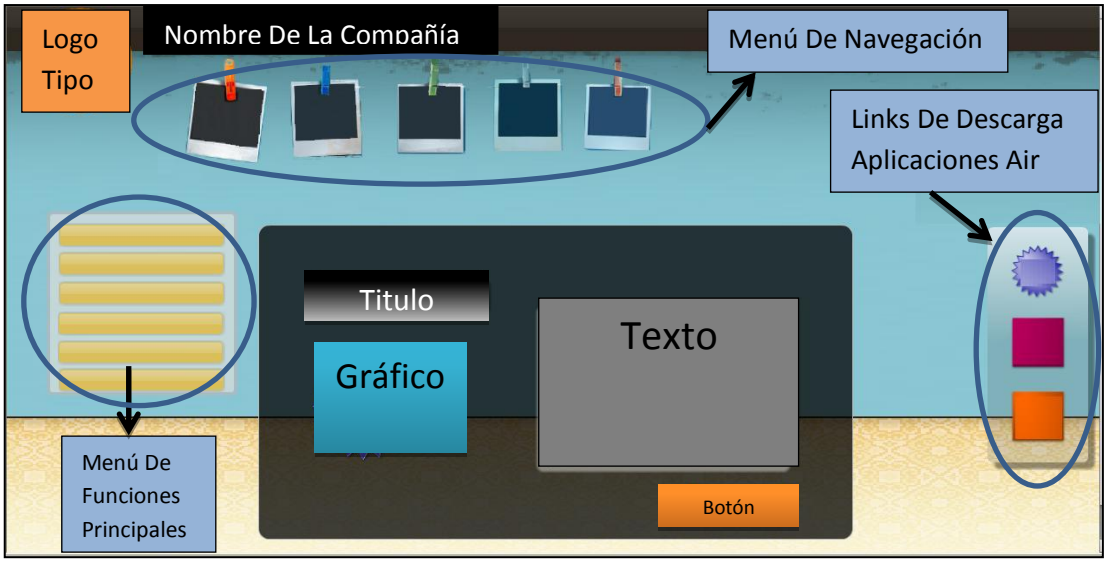


Figura 4.1 Bosquejo de la plantilla de la interfaz del menú general

La interfaz del menú general está diseñada de forma tal que cada usuario pueda acceder de forma sencilla al sitio web. Este cuenta con un menú de navegación con el cuál el usuario puede visitar cada enlace como la misión, visión, contáctenos y acerca de.

Además se cuenta con un menú con funciones específicas para el ingreso de cada usuario, ya sea de tipo administrativo o simplemente sea un usuario de tipo comprador, para lo cual deben identificarse con un usuario y contraseña.

Se ha colocado links de descarga para las aplicaciones Air de escritorio, donde dependiendo el tipo de aplicación que desee descargar se le pedirá que inicie sesión con un usuario y contraseña.

4.3.1. Bosquejos de interfaces según el usuario

El inicio de sesión es similar para todos los tipos de usuarios, ingresan su usuario y contraseña e ingresan a la correspondiente aplicación.

Figura 4.2 Bosquejo de la pantalla de inicio de sesión

4.3.1.1. Interfaz para el usuario de "Quick Delivery"

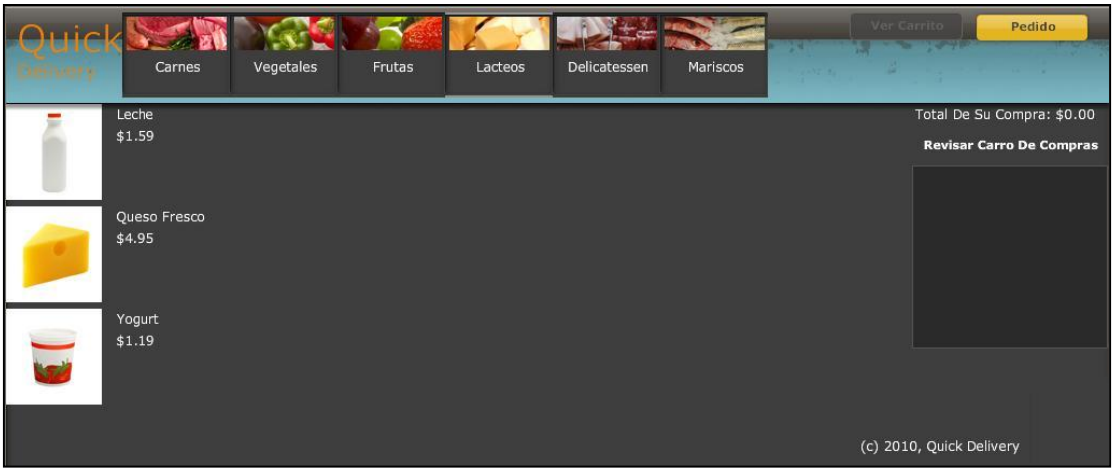


Figura 4.3 Bosquejo de la pantalla Quick Delivery

Figura 4.4 Bosquejo de la pantalla de registro de clientes

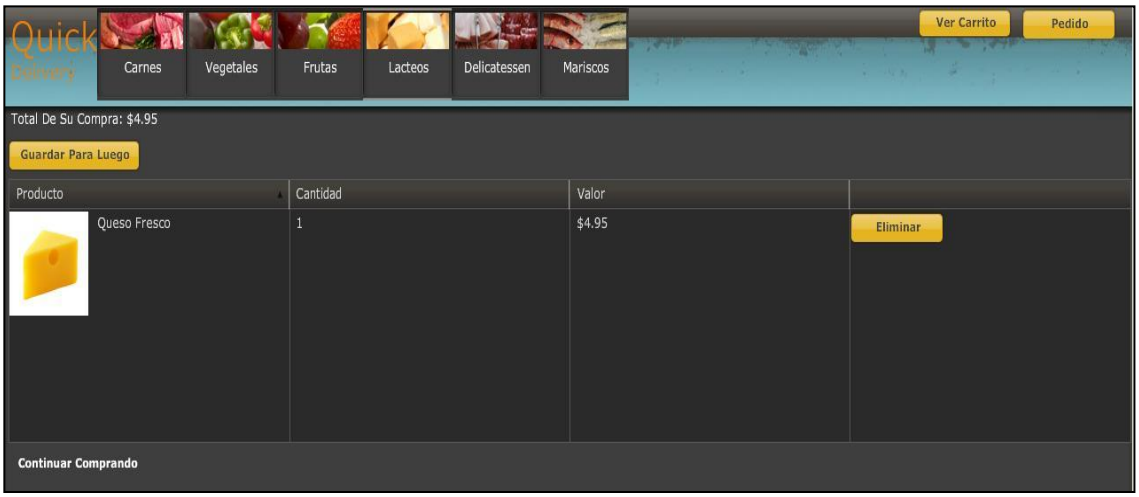


Figura 4.5 Bosquejo de la pantalla del Carro De Compras

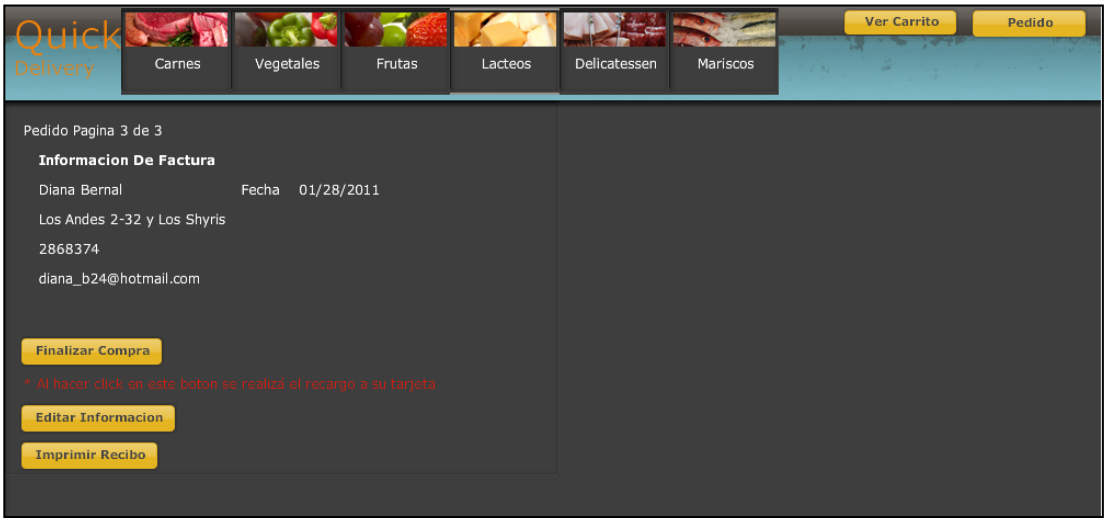


Figura 4.6 Bosquejo de la pantalla final del pedido

4.3.1.2. Interfaz para el usuario de "QEstadìsticas"

Figura 4.7 Bosquejo de la pantalla de QEstadìsticas

4.3.1.3. Interfaz para el mantenimiento de productos

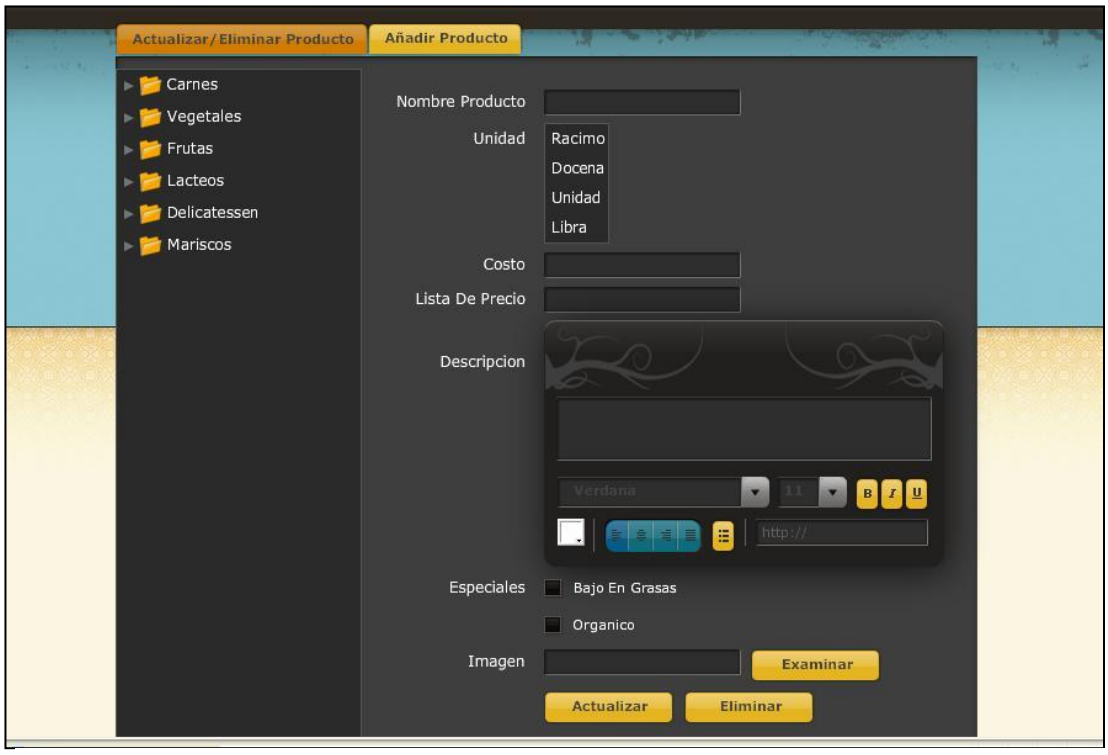


Figura 4.8 Bosquejo de la pantalla de mantenimiento de productos

4.3.1.4. Interfaz para el mantenimiento de usuarios

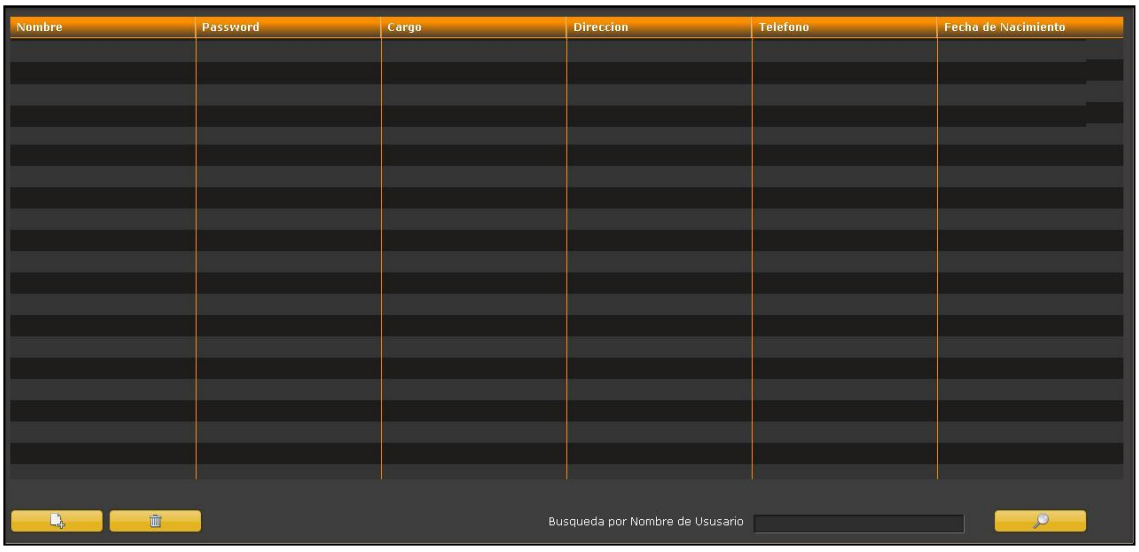


Figura 4.9 Bosquejo de la pantalla de mantenimiento de usuarios

4.4 Diseño de contenido

Se analizará los objetos de contenido y se especificarán los atributos que conforman estos y los mecanismos que se requieren para que establezcan sus relaciones unos con otro, representando la información de un objeto de contenido específico.

Cada atributo incluye información específica de contenido y atributos específicos de implementación que se especifican como parte del diseño.

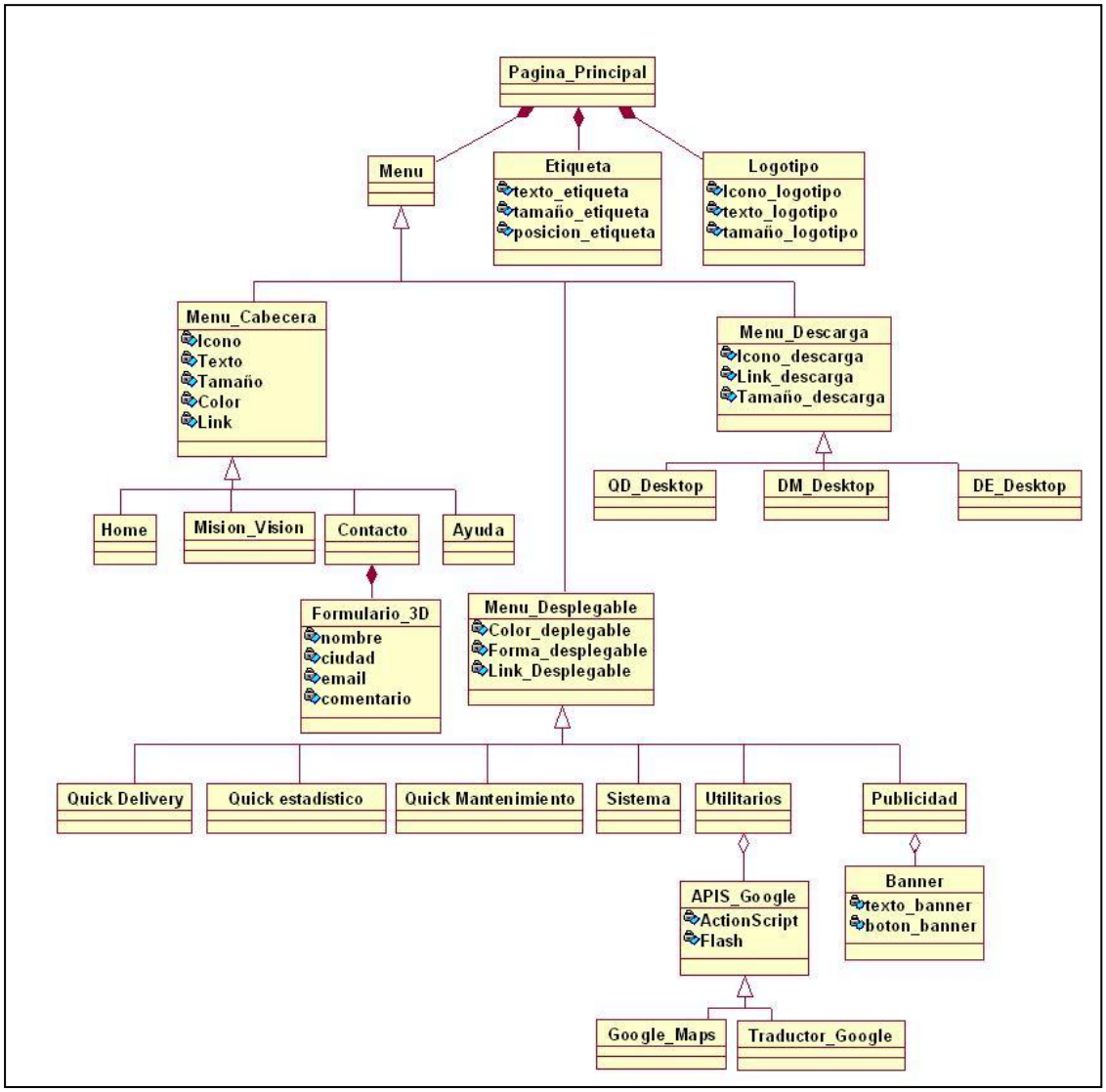


Figura 4.10 Diseño De Contenido Página Principal

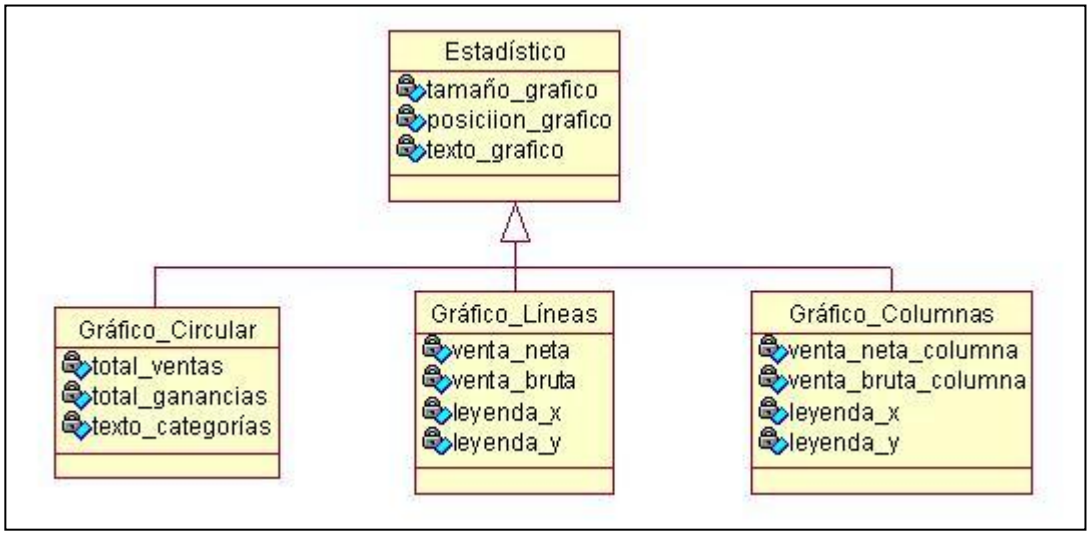


Figura 4. 11 Diseño De Contenido QEstadística

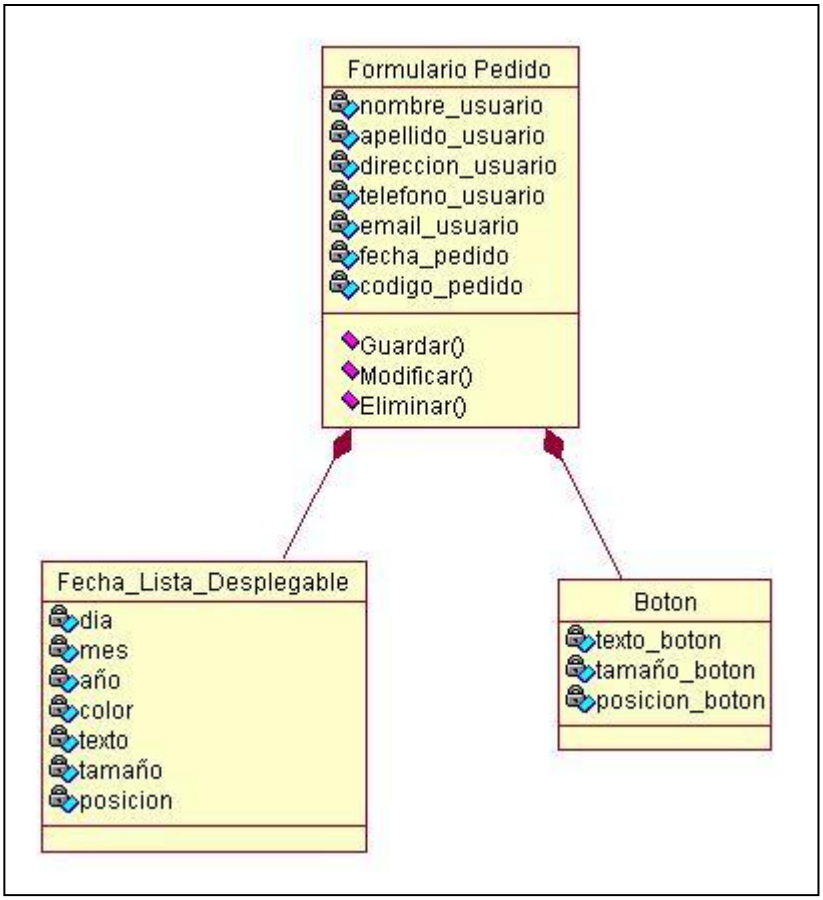


Figura 4.12 Diseño de contenido de formulario

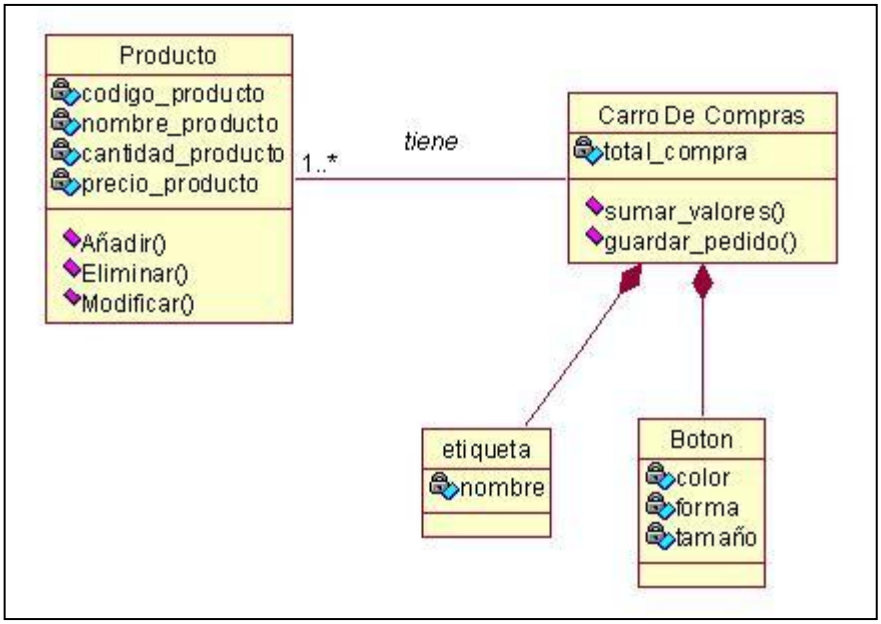


Figura 4.13 Diseño De Contenido Carro De Compras

4.5 Diseño arquitectónico

El diseño arquitectónico está relacionado con las metas establecidas para la Web como el contenido que se presenta, los usuarios que la visitan y el proceso de navegación.

Identificaremos la arquitectura de contenido y la arquitectura de la web

4.5.1 Arquitectura de contenido

La arquitectura de contenido se centra en la forma en la que los objetos de contenido se estructuran para su presentación y navegación, la estructura a utilizarse será la jerárquica

Estructura Jerárquica: Es una de las mejores maneras de organizar la información, además de que es una estructura que se adapta bien a las necesidades del sitio web teniendo en cuenta que nuestra aplicación web partirá de una única página de inicio.

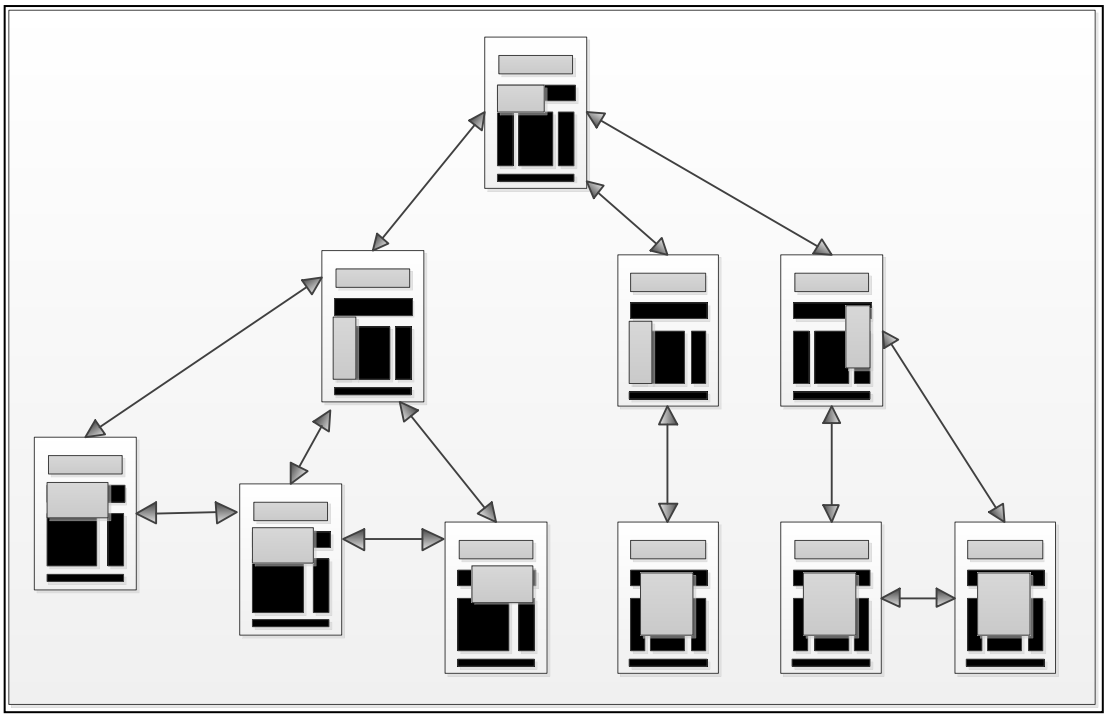


Figura 4.14 Estructura Jerárquica

Estructura Jerárquica Quick Delivery: En nuestra aplicación web partiremos de una única página de inicio, donde se encontrarán los links a las distintas aplicaciones del sistema para los distintos tipos de usuario.

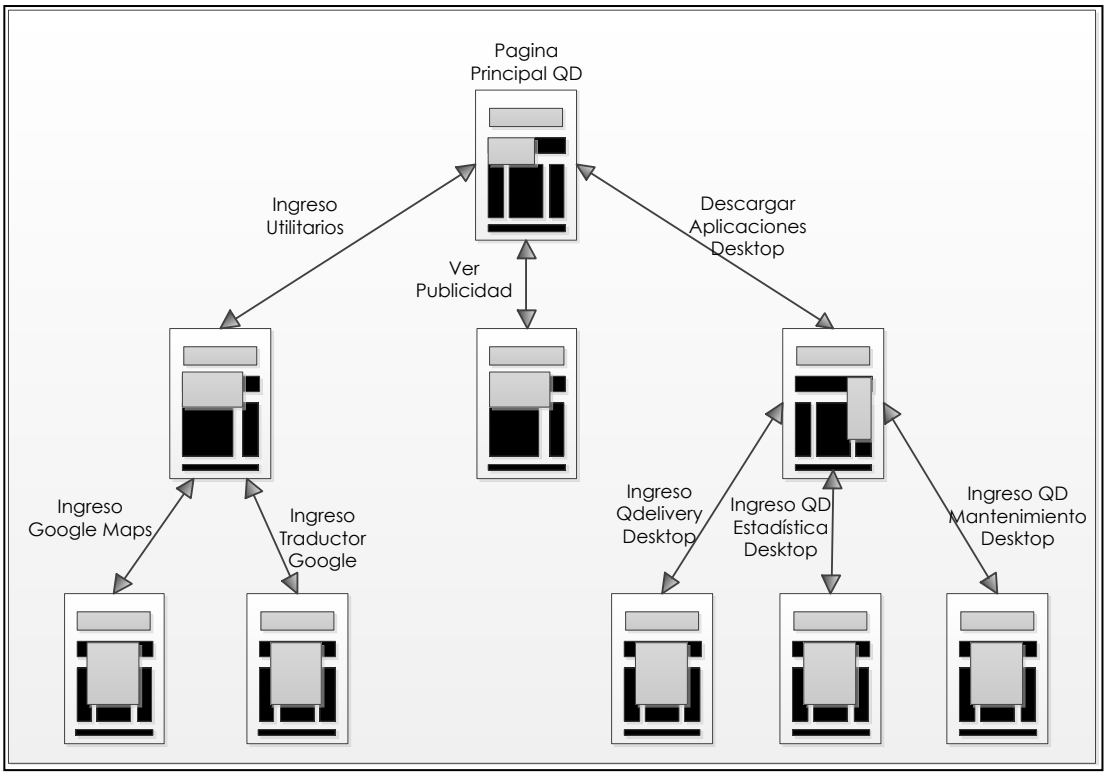


Figura 4.15 Estructura Jerárquica Quick Delivery

Estructura Jerárquica QEstadísticas: El usuario administrador que lleva el seguimiento de las ventas tiene las opciones de ir buscando entre los distintos gráficos estadísticos los datos de las ventas de forma sencilla sin que haya confusión entre los vínculos de cada gráfico.

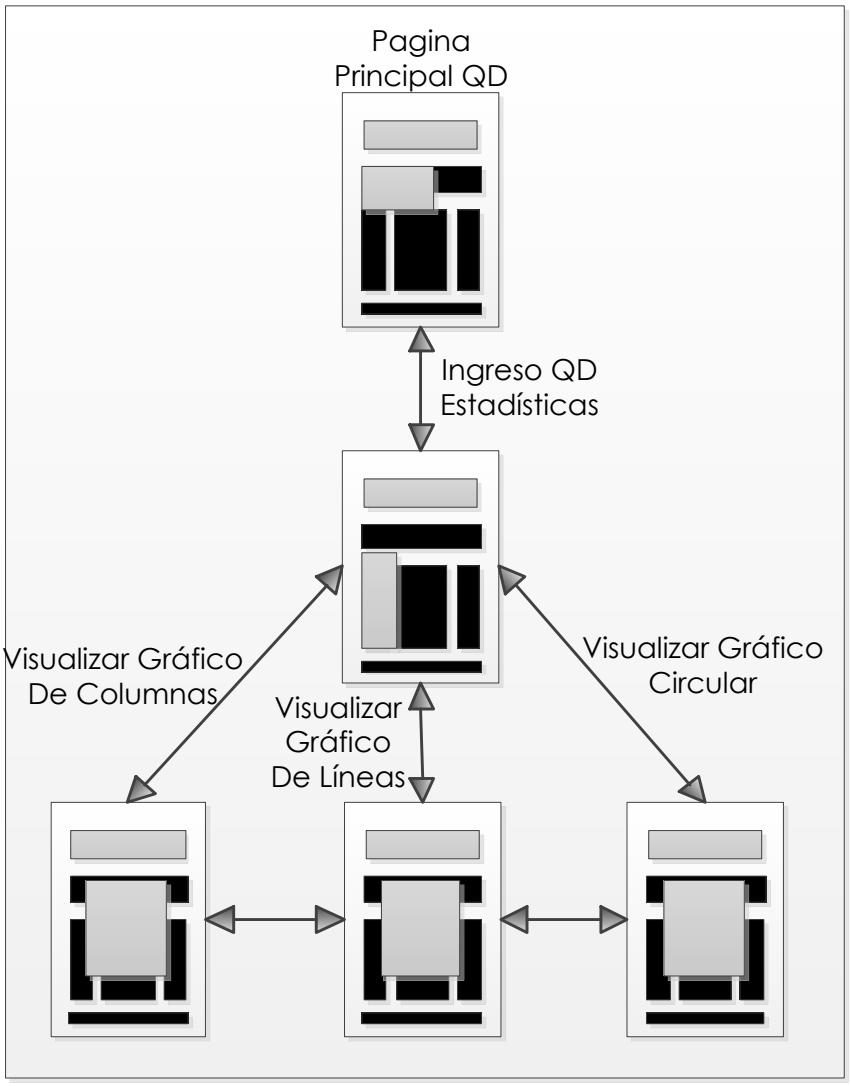


Figura 4.16 Estructura Jerárquica QEstadisticas

Estructura Jerárquica QMantenimiento: El usuario administrador que lleva se encarga del mantenimiento de los productos tiene a su disposición pestañas que lo dirigirán fácilmente a cada una de las opciones de mantenimiento.

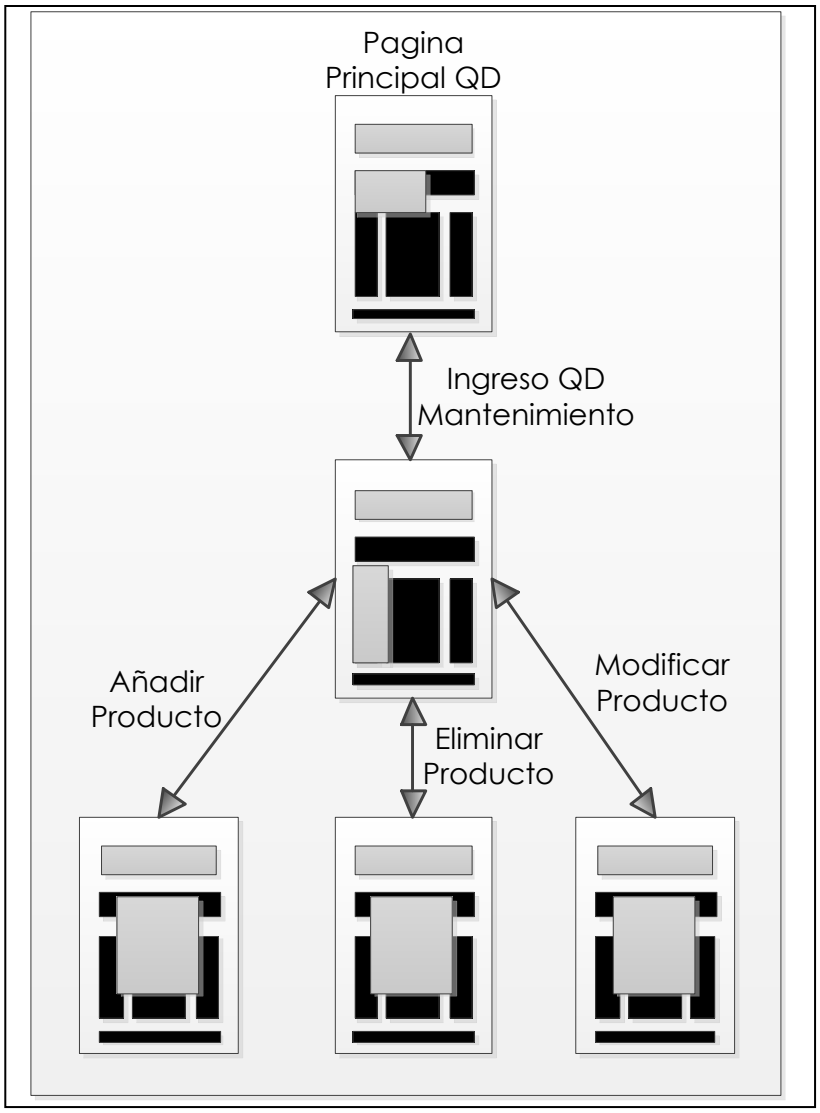


Figura 4. 17 Estructura Jerárquica QMantenimiento

Estructura Jerárquica Para Comprar: El usuario puede ir buscando los productos a través de las categorías disponibles, y cuando decida puede ingresar al carro de compras realizar el pedido e ingresar los datos de la compra, siempre se mantendrá la perspectiva de que no ha cambiado de un lado a otro ya que solamente se actualizan los datos porque no existe recarga de páginas, además de mantener el diseño en todo el proceso de compras.

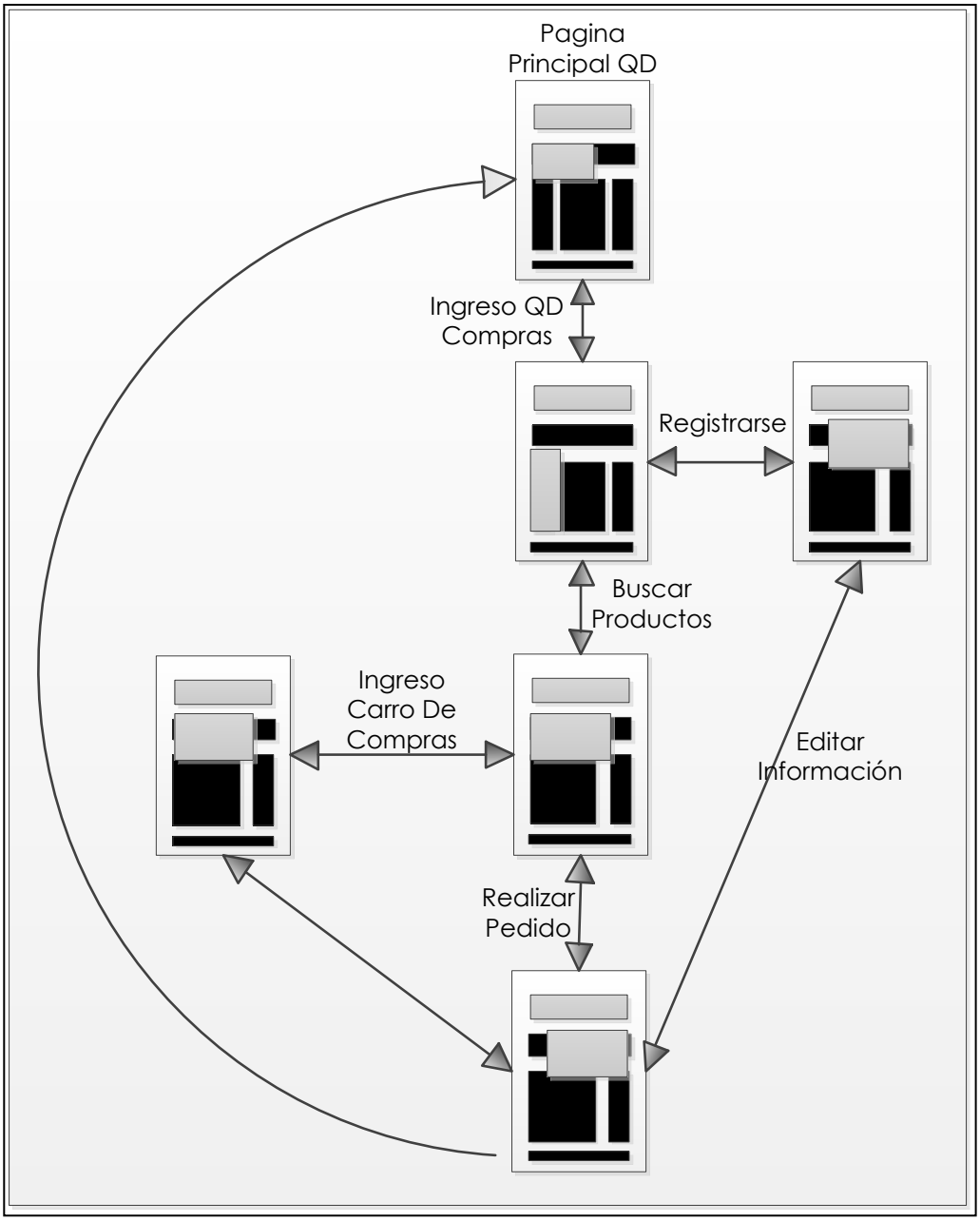


Figura 4.18 Estructura Jerárquica Para Comprar

4.5.2 Arquitectura de la WebApp

La arquitectura empleada en nuestra WebApp es la multicapa. Este tipo de arquitectura permite que el software de una aplicación se subdivida en tres capas:

- **Lógica de presentación**: Contiene todo lo relativo a la presentación (ventanas, gráficos, textos, sonidos, video) hacia el usuario y toda la interacción con el mismo a través de teclado, ratón y micrófonos, etc. Es lo que el usuario percibe de la aplicación. Normalmente se ejecuta en un PC en la mesa del usuario.

- **Lógica de aplicación**: Contiene los algoritmos, procesos y 'workflows' de la aplicación. Es la esencia de la aplicación propiamente dicha.

- **Lógica de datos**: Gestiona todo lo relativo al almacenamiento y recuperación de datos.

La lógica de presentación se ejecuta en un PC en la mesa del usuario, mientras que la lógica de aplicación y la lógica de datos se ejecutan en servidores. En general, cada una de estas capas se ejecuta en máquinas diferentes. Sin embargo, la subdivisión en capas es una arquitectura lógica, no física.

En las aplicaciones web modernas también se tienen clientes que por lo general son un **navegador** (Explorer, Firefox, etc) del que se dispone en todos los PCs (u otro tipo de máquinas) de los usuarios.
El acceso a la aplicación, la obtención de páginas, el almacenamiento y ejecución de AS3, ColdFusion se realiza en el

servidor web que actúa, por decirlo de alguna manera, de servidor de presentación.

En caso de aplicaciones de cierta complejidad, la lógica de aplicación se ejecuta en un servidor de aplicación, implementado como otro servidor web o como un servidor de aplicación propiamente dicho como PHP. Finalmente, se dispone un servidor para datos donde se ejecuta la base de datos como MySQL, etc (Figura 4.19).

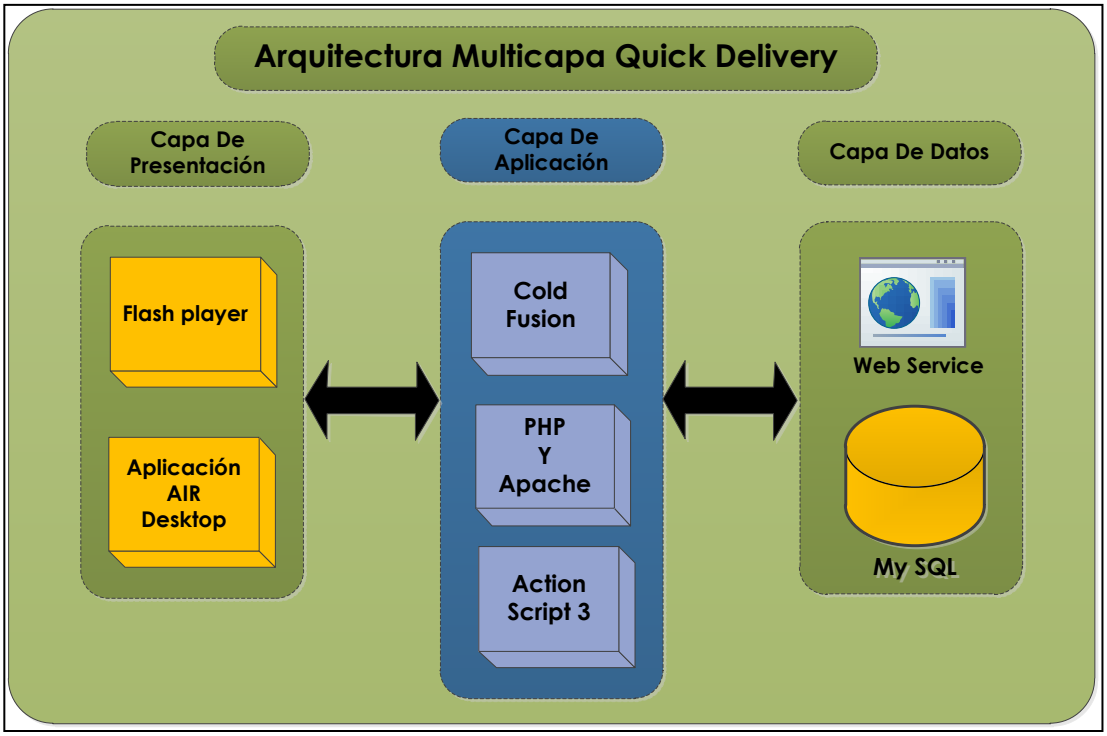


Figura 4.19 Arquitectura Multicapa Quick Delivery

4.5.2.1 Modelo MVC

En las aplicaciones que estamos construyendo pondremos en práctica una forma simple de la arquitectura MVC *(model-view-controller,* Modelo Vista Controlador).

MVC es un patrón de diseño o una arquitectura de software que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres grupos distintos. El objetivo es implementar la lógica para que puedan realizarse los cambios en una parte de la aplicación con un impacto mínimo en las otras partes.

A continuación aparece una breve definición de los términos clave:

- ***Model* (modelo):** Los datos que utiliza la aplicación. Gestiona los elementos de los datos, responde a las peticiones sobre su estado y a las instrucciones para cambiar los datos.

- ***View* (vista):** La interfaz de usuario. Es responsable de presentar los datos del modelo al usuario y de recoger la información del usuario.

- ***Controller* (controlador):** Responde a los eventos, normalmente a los eventos del usuario pero también a los del sistema. Los eventos se interpretan y el controlador invoca cambios en el modelo y en la vista.

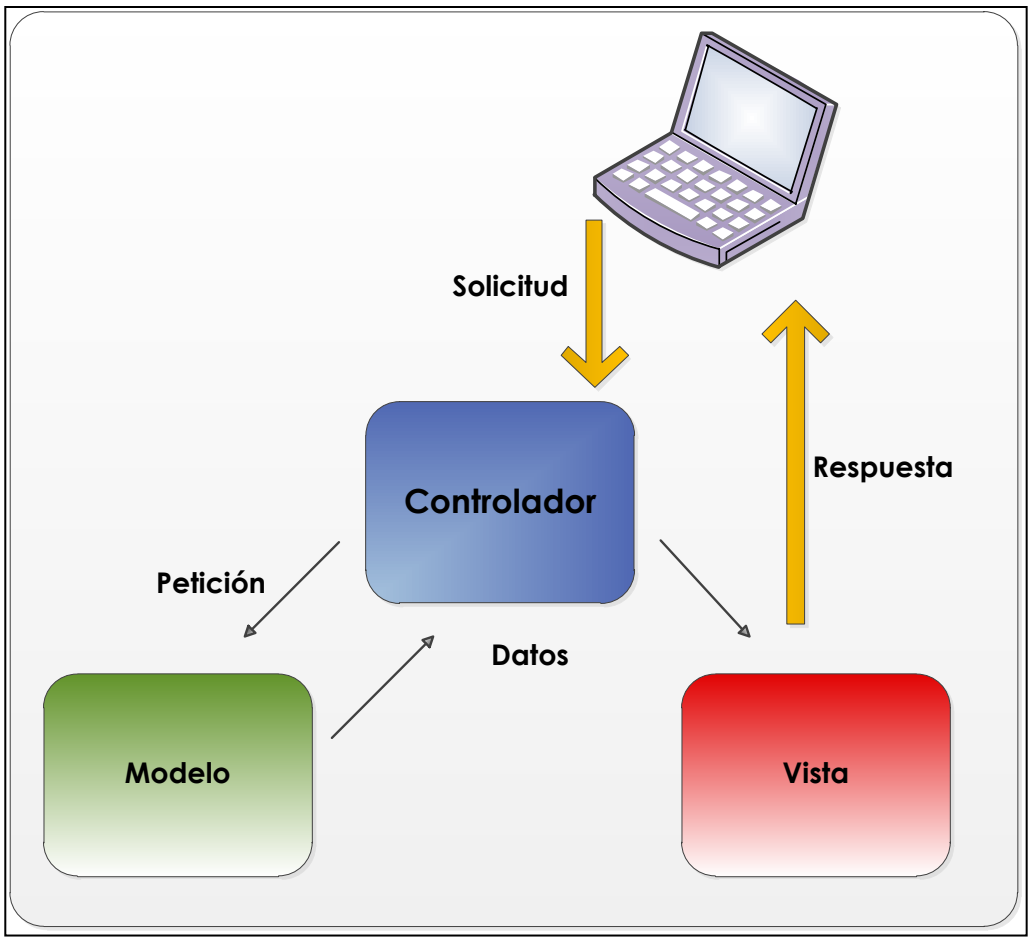


Figura 4. 20 Modelo MVC

En general el proceso de MVC es el siguiente:

1. El usuario interactúa con la interfaz de usuario (una vista) cuando hace clic en un botón para añadir un artículo al carro de la compra.
2. El controlador gestiona el evento que entra.
3. El controlador accede al modelo, bien recuperando o modificando datos.
4. Una vista utiliza entonces los datos del modelo para una representación adecuada al usuario.

Para especificar lo que hará nuestra aplicación principal Quick Delivery tenemos:

- Visualizar las distintas categorías de los artículos comestibles.
- Visualizar los artículos del carro de compras.
- Visualizar una vista detallada de un artículo comestible en concreto.
- Visualizar todos los artículos comestibles en una determinada categoría.

De todas estas acciones estará al frente el controlador, que en nuestro caso es la página Principal QuickDelivery.mxml.
El modelo empezará como una etiqueta <mx:Model> y al final se transformará en datos recuperados de una base de datos.
Ahora que ya tenemos el escenario empezaremos a construir componentes y mejorar la arquitectura y la funcionalidad de la aplicación que estamos construyendo.

4.6 Diseño De Navegación

El diseño de navegación está relacionado con el flujo de navegación del usuario y los objetos que sobresalen dentro de la webApp.

El propósito de la navegación es que el usuario pueda moverse entre una secuencia de objetos de forma intuitiva sin la necesidad de instrucciones de navegación. La información debe ser lo más clara posible para que siempre sepa dónde está, siendo fácil de navegar.

En nuestra AppWeb se ha implementado varios mecanismos para la navegación web tales como:

- Vistas de Estados (ViewStack)
- Menú Vertical y Horizontal
- Vínculos gráficos
- Botones de ingreso
- Botones para descargas
- Vínculos de texto
- Listas
- TabNavigator
- Deep Linking (enlace profundo)

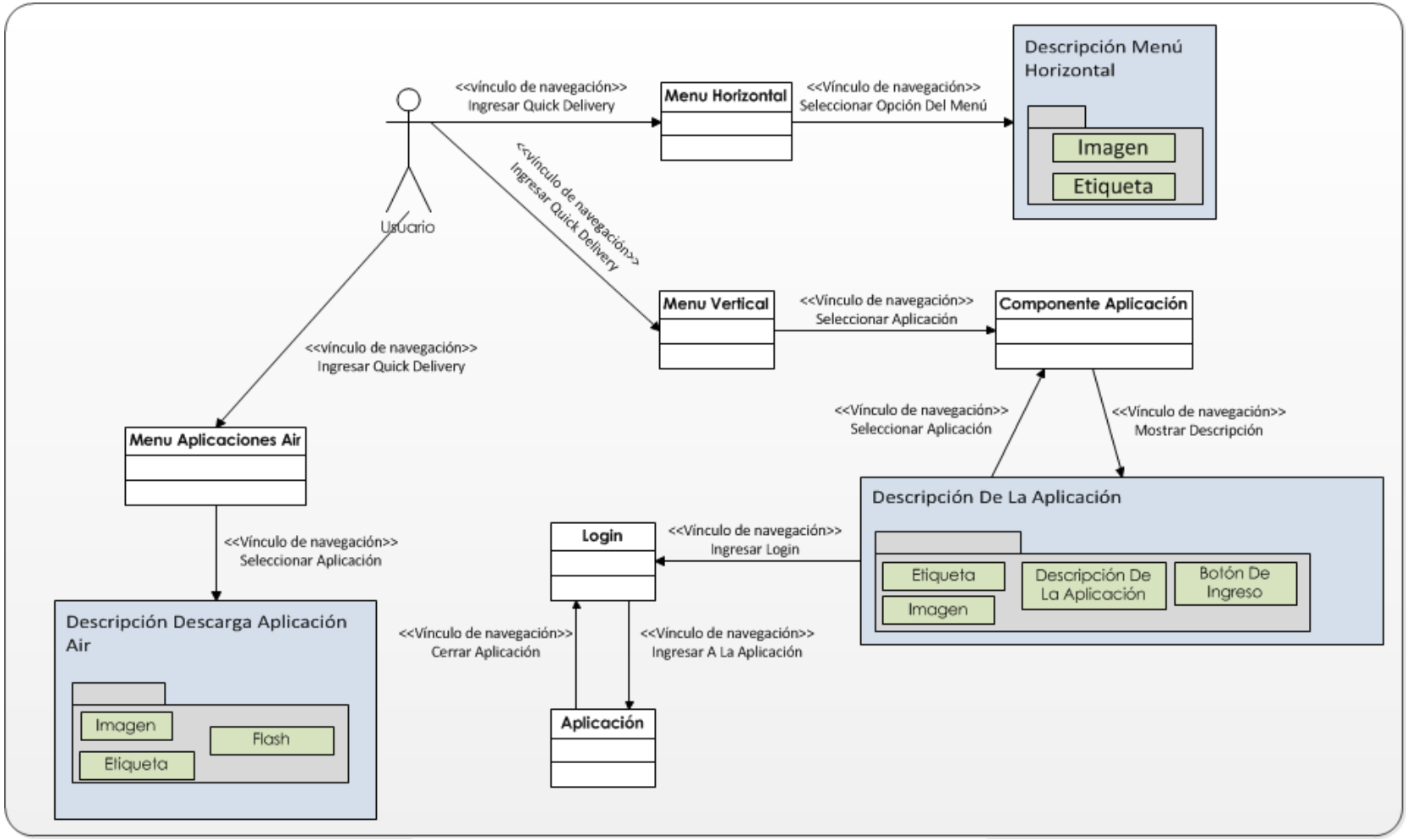


Figura 4. 21 Semántica De Navegación QDelivery

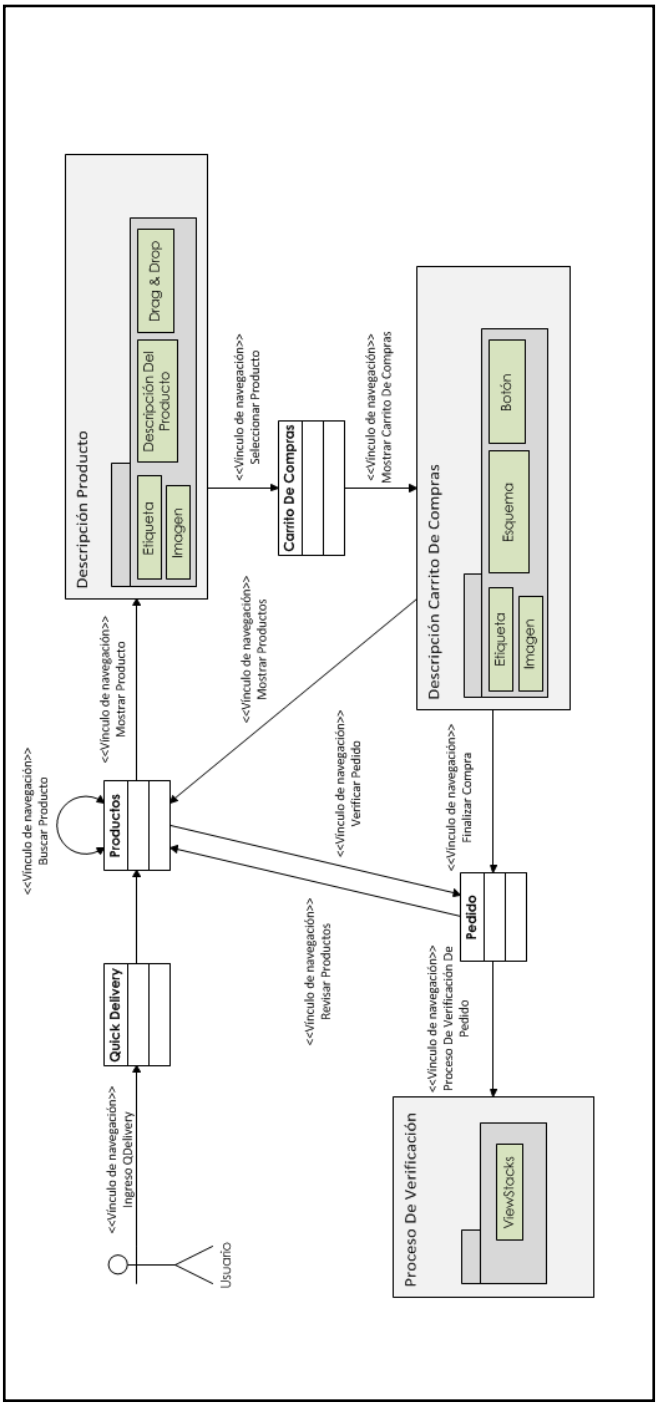


Figura 4. 22 Semántica De Navegación Compras

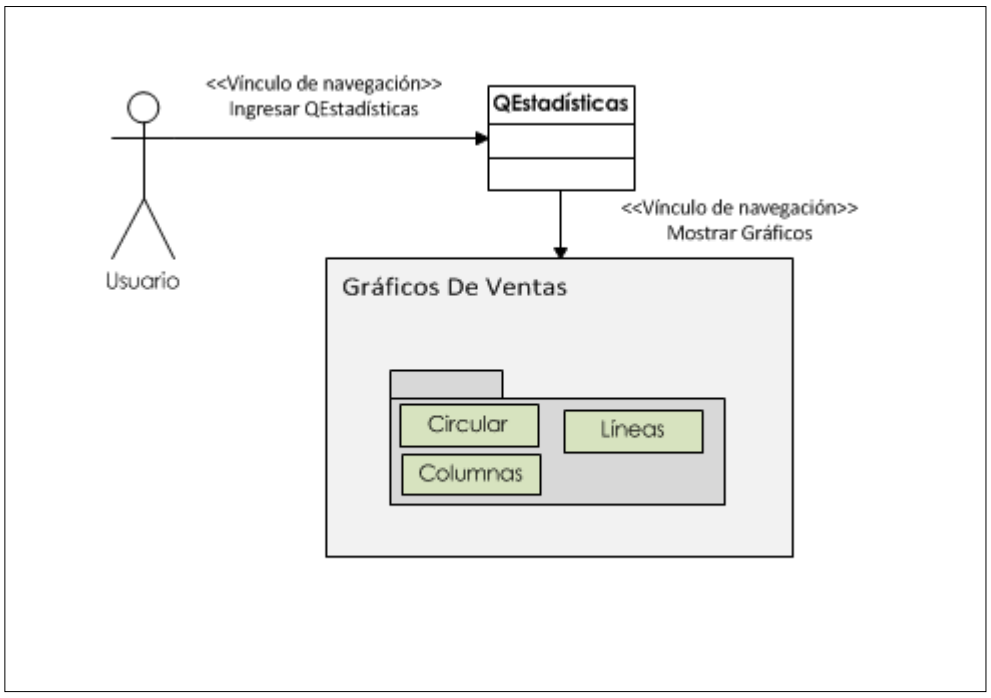


Figura 4. 23 Semántica De Navegación QEstadísticas

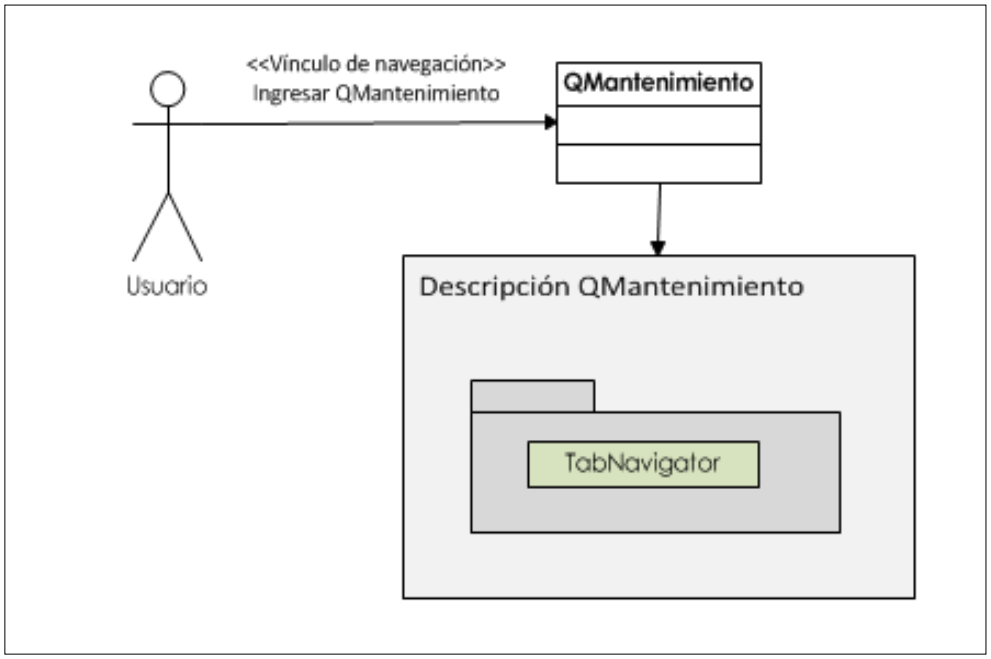


Figura 4. 24 Semántica De Navegación QMantenimiento

4.7 Conclusión

Durante la realización del diseño de las interfaces del sistema, hemos podido comprobar que un diseño web efectivo permite que los distintos usuarios del sitio web puedan acceder con mayor facilidad a los contenidos e interactuar con eficacia con los componentes que añadimos en el sitio; haciendo que los usuarios se familiaricen con el funcionamiento de este y se sientan cómodos y confiados al utilizar nuestras aplicaciones.

Hay que resaltar que gracias a la implementación de la arquitectura MVC hemos logrado crear aplicaciones independientes que interactúen entre sí como un todo, pero que al mismo tiempo nos permite realizar mejoras o cambios en el sistema de forma local sin que esto afecte a los demás componentes ni al diseño visual causando el menor impacto a los usuarios.

Estamos seguros que el éxito o fracaso de nuestra AppWeb depende de un correcto diseño.

CAPÍTULO V

CAPÍTULO 5

GESTIÓN DE RIESGOS

5.1. Introducción

En este capítulo se realizará la identificación de potenciales riesgos que se pueden dar en el transcurso del desarrollo de la aplicación, para evitar que los mismos provoquen retrasos en lo estimado previamente causando pérdida de tiempo y dinero.

Como parte de este plan de contingencia de riesgos se realizará el análisis de cada uno de estos con la finalidad de saber cómo evitarlos y superarlos sin que esto conlleve perdidas en el proyecto.

Muchas de las principales causas de los problemas dentro del desarrollo de un sistema que se da en el entorno informático es la inadecuada gestión de riesgos y por esta razón la necesidad de gestionar una adecuada administración de los riesgos. Para esto se llevará a cabo un análisis y priorización de los riesgos para posteriormente desarrollar un plan de mitigación de los riesgos.

5.2. Identificación de los riesgos

Riesgos en cuanto al personal: Un miembro del proyecto abandona el proyecto antes de su finalización o no mantiene el ritmo de su trabajo afectando así el desarrollo del proyecto en cuanto a tiempo de desarrollo.

Interfaz: Esta se relaciona a las verificaciones que aseguren la validez y la completa información introducida dentro del sistema y así mismo que la información ha sido procesada y transmitida adecuadamente por las aplicaciones.

Administración de cambios: Este tipo de riesgo se da cuando no se ha comunicado al usuario final de los cambios realizados en el sistema, como cambios en los procesos y la forma de implementarlos en el sistema.

Información: Estos riesgos están asociados con la integridad de la seguridad de la información procesada y la administración efectiva de los sistemas de bases de datos. La integridad puede perderse por errores de programación, consultas o transacciones mal realizadas o por la falta de mantenimiento del sistema.

Riesgos de acceso: Estos riesgos se enfocan al inapropiado acceso a sistemas, datos e información:

- **Aplicación:** La aplicación no da la seguridad a los usuarios sobre las funciones necesarias para ejecutar su trabajo.
- **Información:** El mecanismo no provee a los usuarios acceso a la información específica del entorno.
- **Entorno de procesamiento:** Estos riesgos en esta área están manejados por el acceso inapropiado al entorno de programas e información.
- **Redes:** En esta área se refiere al acceso inapropiado al entorno de red y su procesamiento.

Riesgos en la infraestructura: Estos riesgos se deben a la falta de una estructura tecnológica efectiva (hardware, software, redes, personas y procesos) para soportar adecuadamente las necesidades futuras y presentes del sistema. Para esto se consideran los siguientes peligros:

- Falta de Backups y planes de contingencia para controlar los desastres en el procesamiento de la información.
- No existen técnicas de recuperación/restauración en caso de pérdida de información.

Operaciones de red y computacionales: Se pueden ver afectados los sistemas de información y los entornos de red cuando no están operados en un esquema seguro y protegido, las responsabilidades de procesamiento de la información no están siendo ejecutadas por personal operativo definido y monitoreado. También hay que asegurarse que los sistemas sean consistentes y estén siempre disponibles a los usuarios a un nivel de ejecución satisfactorio.

Administración de la base de datos: Se tiene que asegurar que las bases de datos usadas en caso de una situación peligrosa tenga soporte para las posibles operaciones críticas que se puedan presentar en el procesamiento de la información.

Riesgos en la creación de la planificación:

- EL tiempo para el desarrollo del sistema no es el adecuado y un proyecto de tal magnitud no se puede terminar en el tiempo asignado.
- El esfuerzo es mayor que el estimado (por líneas de código, puntos de función, módulos, etc.)
- La presión excesiva en la planificación reduce la productividad.

- La fecha final ha cambiado sin ajustarse al ámbito del producto o a los recursos disponibles.
- Tener tareas desconocidas del proyecto llevan más tiempo del esperado en el diseño e implementación.

Riesgos en el entorno de desarrollo:

- Los espacios no están disponibles en el momento necesario, o no son adecuados.
- Las herramientas de desarrollo no funcionan como se esperaba.

Riesgos en cuanto a los usuarios finales:

- Los usuarios finales no tienen conocimientos técnicos sobre el proyecto.
- Los usuarios finales insisten en nuevos requerimientos.
- Al final a los usuarios no les gusta el producto desarrollado, hay que volver a diseñarlo y construirlo.
- Los usuarios no entiende el funcionamiento de la página web.

Riesgos en cuanto a los requisitos:

- Los requisitos se han adaptado, pero continúan cambiando.
- Los requisitos abarcan otros sistemas no probados su funcionamiento.
- Se añaden requisitos extras, o no se han definido correctamente.

Riesgos en cuanto al producto:

- La calidad no es aceptable y requiere de un trabajo de comprobación, diseño e implementación superior al esperado.
- El diseño de una interfaz de usuario no es la adecuada y requiere volver a diseñarla e implementarla.
- El trabajo con un entorno no conocido causa problemas imprevistos.

- Las personas claves solo están disponibles una parte del tiempo.
- El personal trabaja más lento de lo esperado.

Riesgos en cuanto al diseño y la implementación:

- Diseño muy sencillo.
- El diseño puede ser demasiado complejo y tener complicaciones innecesarias e improductivas en la implementación.
- Un mal diseño implica volver a diseñar e implementar.
- Los componentes desarrollados por separado no se pueden integrar de forma sencilla.

5.3. Análisis de los riesgos

El primer paso en el análisis es el evaluar los riesgos. Esto involucra comparar el nivel de riesgo detectado durante el proceso de análisis con los siguientes criterios: probabilidad, magnitud de pérdida y exposición al riesgo.

Tabla de la evaluación de los riesgos

RIESGO	Probabilidad	Magnitud De Pérdida (Días)	Exposición Al Riesgo (Días)
Cambio de requisitos	0,05	3	0.15
Escatimar en la calidad	0,2	5	1.0
Diseño inadecuado	0,15	1	0.15
Desarrollo orientado a la investigación	0,25	4	1.0
Planificación demasiado optimista	0,2	2	0.4
El tiempo requerido por una tarea puede superar al tiempo estimado	0,1	4	0.4
Actividades no contempladas a tiempo atrasan el proyecto	0,2	2	0.4

Falta de dominio en el manejo de la herramienta	0,05	3	0.15
El programa puede no cumplir las expectativas requeridas por el usuario (Interfaz, facilidad de uso, etc.)	0,15	1	0.15
El cliente desconoce el tiempo real de planificación y desarrollo	0,1	1	0.1
Disponibilidad de tiempo entre los desarrolladores	0,5	5	2.5
La falta de motivación y de moral reduce la productividad	0,2	2	0.4
Falta de integración en los módulos programados	0,1	3	0.3

Tabla 1 Evaluación de los riesgos

5.4 Priorización de los riesgos

Una vez realizado el análisis de los riesgos con base a los aspectos de probabilidad e impacto, es recomendable utilizar una matriz de priorización que permite determinar cuales requieren un esfuerzo en la gestión de riesgos.

Tabla de estimación de riesgos priorizada

Riesgo	Probabilidad de pérdida	Magnitud de la pérdida (semanas)	Exposición a riesgo (semanas)
Diseño inadecuado.	25%	6	1,5
Subestimar el tiempo de desarrollo.	25%	6	1,5
Conflictos para la adquisición de información externa.	15%	2	0,3
Subestimar el sistema de la red existente.	15%	2	0,3
Escatimar en la calidad	10%	3	0,3
Planificación demasiado optimista.	5%	2	0,1
Síndrome de la panacea.	5%	2	0,1

Tabla 2 Estimación de riesgos priorizada

5.5 Plan de mitigación

El plan de mitigación de riesgos nos muestra cómo los riesgos específicos serán tratados y de las medidas de acción que son necesarios para llevarlas a cabo. Con esto se tiene en claro qué acciones se están tomando para mejorar el riesgo del proyecto.

Planificación de la gestión de riesgos

Riesgo	Plan
Planificación demasiado optimista	Utilizar un cronograma, con las actividades **básicas** bien definidas.
Disponibilidad de tiempo entre los desarrolladores	Fijar turnos para los desarrolladores y reuniones periódicas para coordinar y controlar las tareas establecidas en el cronograma.
Problemas con el personal	Contratar y planificar los miembros clave del equipo mucho antes de que comience el proyecto
	Tener buenas relaciones con el personal contratado
Escatimar en la calidad	Fijar lineamientos y estándares de desarrollo que permita tener un producto de calidad sin profundizar en detalles irrelevantes

Diseño inadecuado	Establecimiento de requerimientos puntuales del software por parte del cliente desde el inicio del proyecto.
	Uso de modelos prototipo
El tiempo requerido por una tarea puede superar al tiempo estimado	Estimar correctamente los tiempos en el cronograma, dejando holguras suficientes para cubrir cualquier imprevisto.
Falta de dominio en el manejo de la herramienta	Soporte por parte de personas que dominan la herramienta.
Acceso ilegal a los servicios informáticos	Utilización de claves y contraseñas para permitir el acceso a los equipos
Riesgos producidos por las diferentes clases de virus y programas destructivos que existen	Utilización de antivirus y firewall
Acceso ilegal a la información	Encriptar y la desencriptar la información manipulada, de forma que sólo las personas autorizadas pueden acceder a ella

Tabla 3 Planificación de la gestión de riesgos

5.6 Conclusión

En este capítulo hemos identificado algunos de los riesgos más comunes a los que un proyecto Web está expuesto, y este en caso particular, a la vez hemos analizado y determinado cual será el posible tratamiento que se le debe dar a cada uno de estos procurando poner mayor atención en aquellos que los hemos calificado como inminentes para con esto conseguir eliminar las fuentes de riesgo antes de que estos empiecen a afectar al cumplimiento de los objetivos del proyecto.

CAPÍTULO VI

CAPÍTULO 6

DESARROLLO DE UN SISTEMA PARA COMERCIO ELECTRÓNICO DE ENTREGAS Y REPARTOS "QUICK DELIVERY"

6.1 Introducción

Una vez que ya se ha analizado y profundizado en lo que respecta a la tecnología RIA y sus herramientas podemos dar paso al desarrollo de una aplicación RIA orientada al comercio electrónico, para la misma que ya hemos realizado el respectivo análisis para su correcta codificación y desarrollo.

El sistema que presentaremos a continuación funcionará como un sitio web de comercio electrónico de entregas y repartos a la que llamaremos Quick Delivery, en la que ofreceremos productos alimenticios de carácter perecible, los mismos que deberán ser entregados a los usuarios que compren en nuestro sitio web de manera inmediata.

Este sistema será desarrollado sobre la plataforma Eclipse, usando una herramienta llamada Adobe Flex Builder 3 de la casa comercial de Adobe, del cual poseemos licencia original Estudiantil, a esta herramienta se añadirán programas como los mencionados en capítulos anteriores como Adobe AIR (aplicación Desktop), Adobe Flash Player, un servidor Web para el consumo de Web Services llamado ColdFusion, Servidor local Apache con el uso de PHP y MySQL para el manejo de base de datos.

A la par con estas herramientas usaremos librerías de código abierto para ejemplificar el potencial del desarrollo en 3D como es PaperVision en la tecnología RIA, además el uso de Coulds APIs de Google Language y Google Maps, lo cual estaremos explicando en detalle durante el desarrollo de este capítulo.

6.2. Configuración y demostración del uso de las herramientas

6.2.1. Adobe Flex Builder 3

Flex, como ya lo mencionamos en el capítulo 2 es un marco de trabajo de código abierto gratuito destinado para la creación de aplicaciones web dinámicas que se implantan coherentemente en exploradores, computadores de mesa y sistemas operativos, aprovechando los tiempos de ejecución de Adobe Flash Player y Adobe AIR. Aunque se pueden crear aplicaciones Flex únicamente usando el marco de trabajo de Flex SDK, con Adobe Flex Builder 3 podemos acelerar el desarrollo mediante funciones como códigos inteligentes, depuración interactiva estratificada y diseño visual del aspecto de la interfaz de usuario.

Adobe ofrece algunas opciones de licenciamiento de sus productos, de los cuales nosotros escogimos la licencia Educativa como parte del programa que Adobe promueve para difusión de su tecnología.

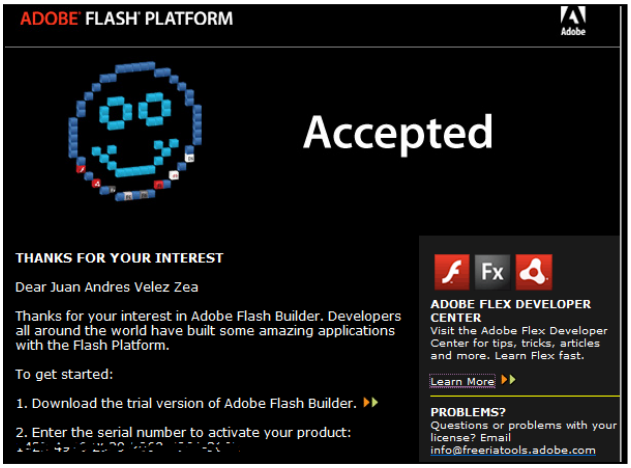


Figura 6. 1 Licencia Adobe Flex Builder 3 Educacional

Aunque actualmente Adobe Flex dispone de la versión Adobe Flex Builder 4 o con su nombre comercial Flash Builder 4 nosotros trabajaremos con la versión 3 de Flex Builder, esto debido a que durante el desarrollo de nuestro trabajo se liberó al mercado esta versión de la cual también poseemos licencia Estudiantil original.

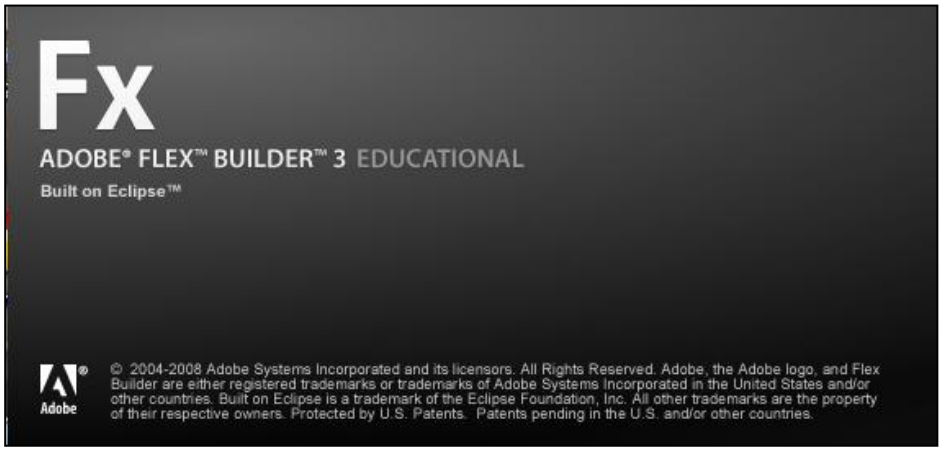


Figura 6. 2 Adobe Flex Builder 3

Es importante recalcar en esta parte, porque gracias a este licenciamiento tenemos la posibilidad de participar en programas de mejora continua y aprendizaje sin limitaciones de ningún tipo en la parte legal.

Una vez que hemos realizado esta explicación vamos a profundizar en el programa como tal, procurando enfocarnos en su estructura y funcionamiento durante el desarrollo de nuestro sistema "Quick Delivery".

6.2.2. Creación de Quick Delivery

Nuestro proyecto Quick Delivery está formado por tres aplicaciones dirigidas a diferentes tipos de usuarios; la primera llamada Quick Delivery el cual es el sitio web de compras para los usuarios en la que estos podrán ver nuestros productos y hacer sus pedidos, la segunda aplicación se llama QD Mantenimiento y está destinada al usuario administrador el cual realizará el mantenimiento de los productos, y la

tercera aplicación llamada QD Estadísticas servirá como bodega de información en la que podremos visualizar los datos estadísticos de ventas de nuestros productos.

Ahora que hemos explicado cual será el contenido de nuestra aplicación RIA procederemos al desarrollo de la misma.

La plataforma eclipse es de código abierto, un entorno de desarrollo integrado (IDE) que puede ampliarse, y Flex Builder ha ampliado y personalizado Eclipse para construir las aplicaciones Flex; lo primero que haremos es crear un nuevo proyecto en FB3.

Seleccionamos **File> New> Flex Project**, como Project Name escribimos Quick Delivery, en Project Location dejamos como Use Default Location, seleccionamos Web Application en Application Type, en la opción Server Technology seleccionaremos PHP ya que uno de nuestros servidores locales es Apache (se explicara más adelante el otro).

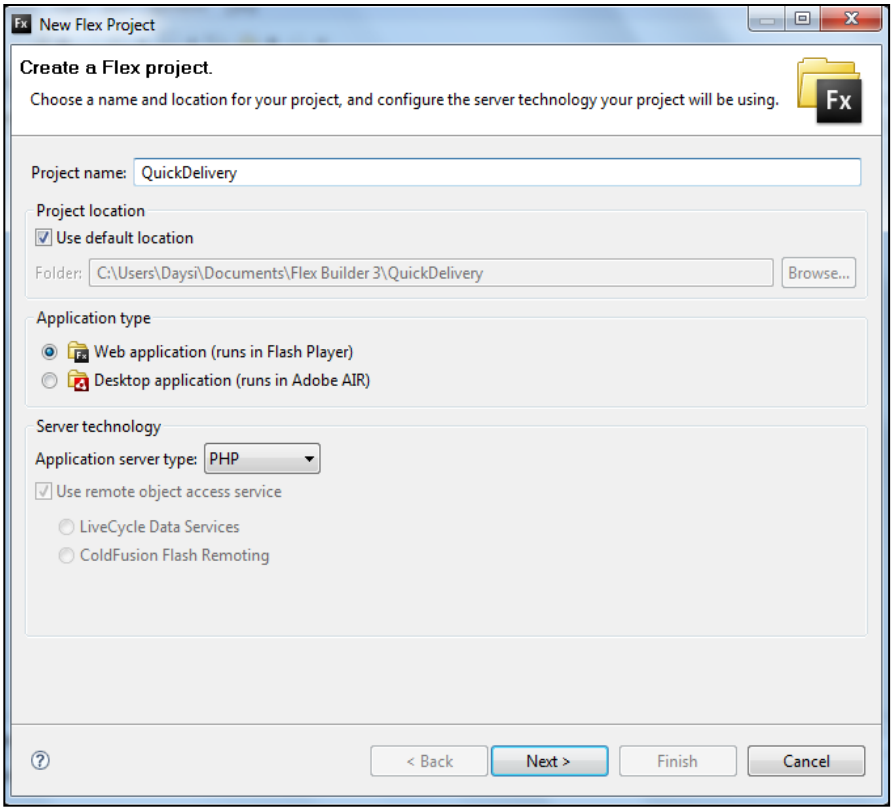


Figura 6. 3 Pantalla de Creación de Proyecto QuickDelivery con Flex 3

En la pantalla siguiente configuramos las opciones del servidor, en este proyecto hemos usado la aplicación WAMP Server con la que se instala el Servidor local Apache versión 2.2.11, PHP 5.2.9-2, y MySQL 5.1.36 como base de datos. Por lo que tenemos que especificar esta información en los campos de esta pantalla y a continuación validamos esta información para continuar.

Web root: C:\wamp\www\phpTesis

Root URL: http://localhost/phpTesis

Output Folder: C:\wamp\www\phpTesis\QuickDelivery-debug

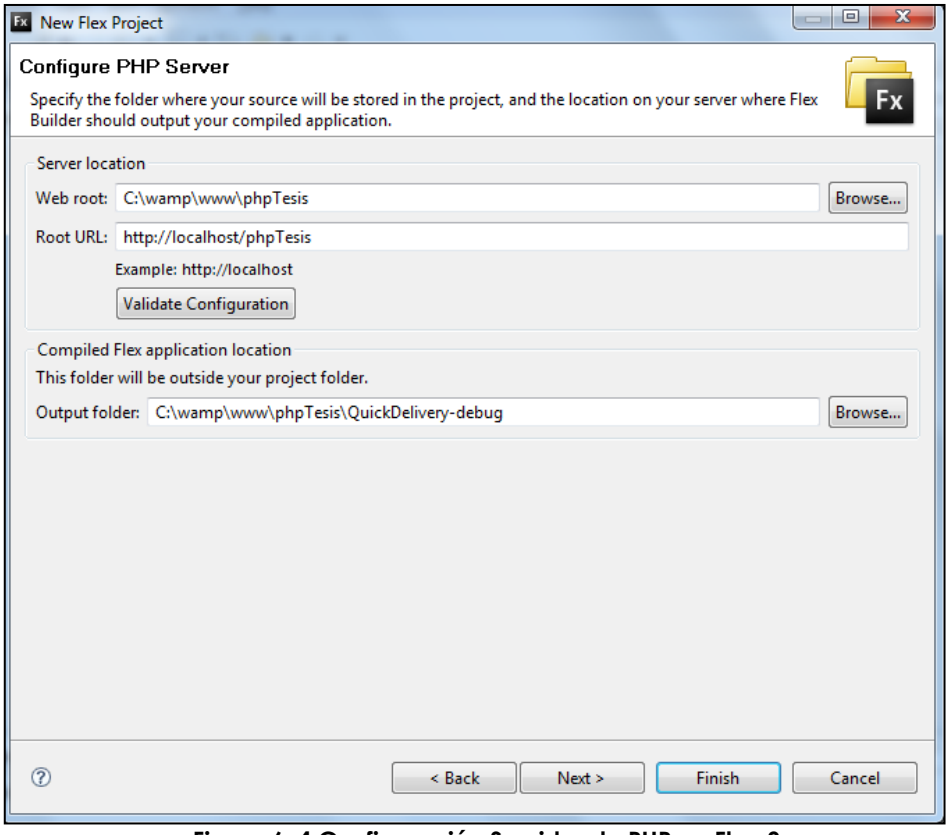


Figura 6. 4 Configuración Servidor de PHP en Flex 3

En la última pantalla configuraremos los parámetros de las librerías, módulos que se usan y como se configura el formato del proyecto, por el momento Flex creará de forma automática el archivo para la aplicación principal e incluirá la estructura básica de un archivo de aplicación Flex.

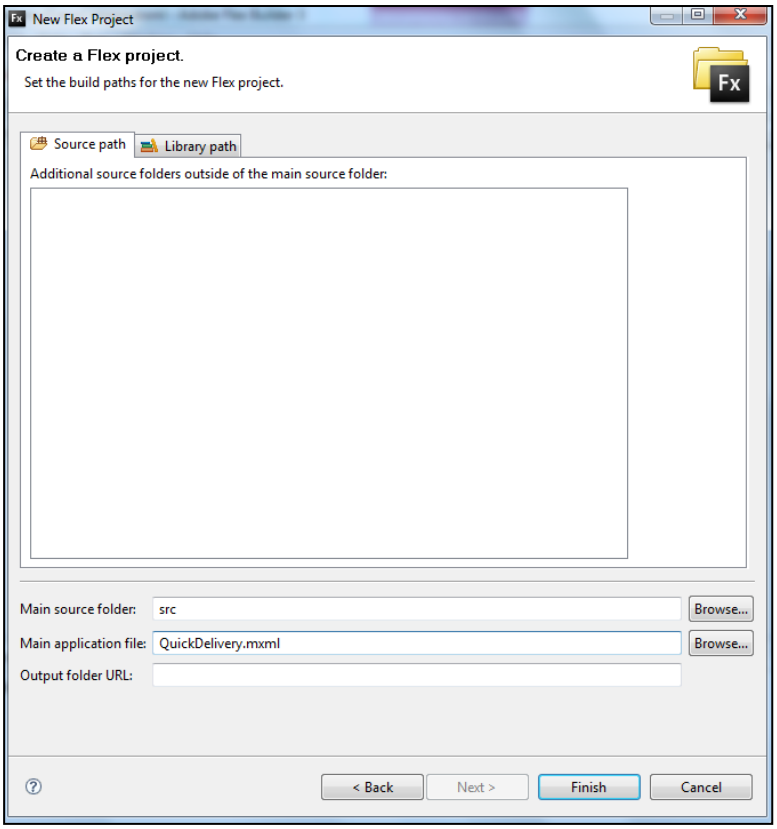


Figura 6. 5 Configuración de Path y Librerías de la Aplicación

Al final hacemos clic en Finish y se configura Flex en su forma de mesa de trabajo (workbench) y veremos creados los archivos del proyecto y la aplicación, cabe indicar que esta aplicación aparecerá en forma SOURCE (código) pero podemos cambiar la perspectiva a DESING (diseño).

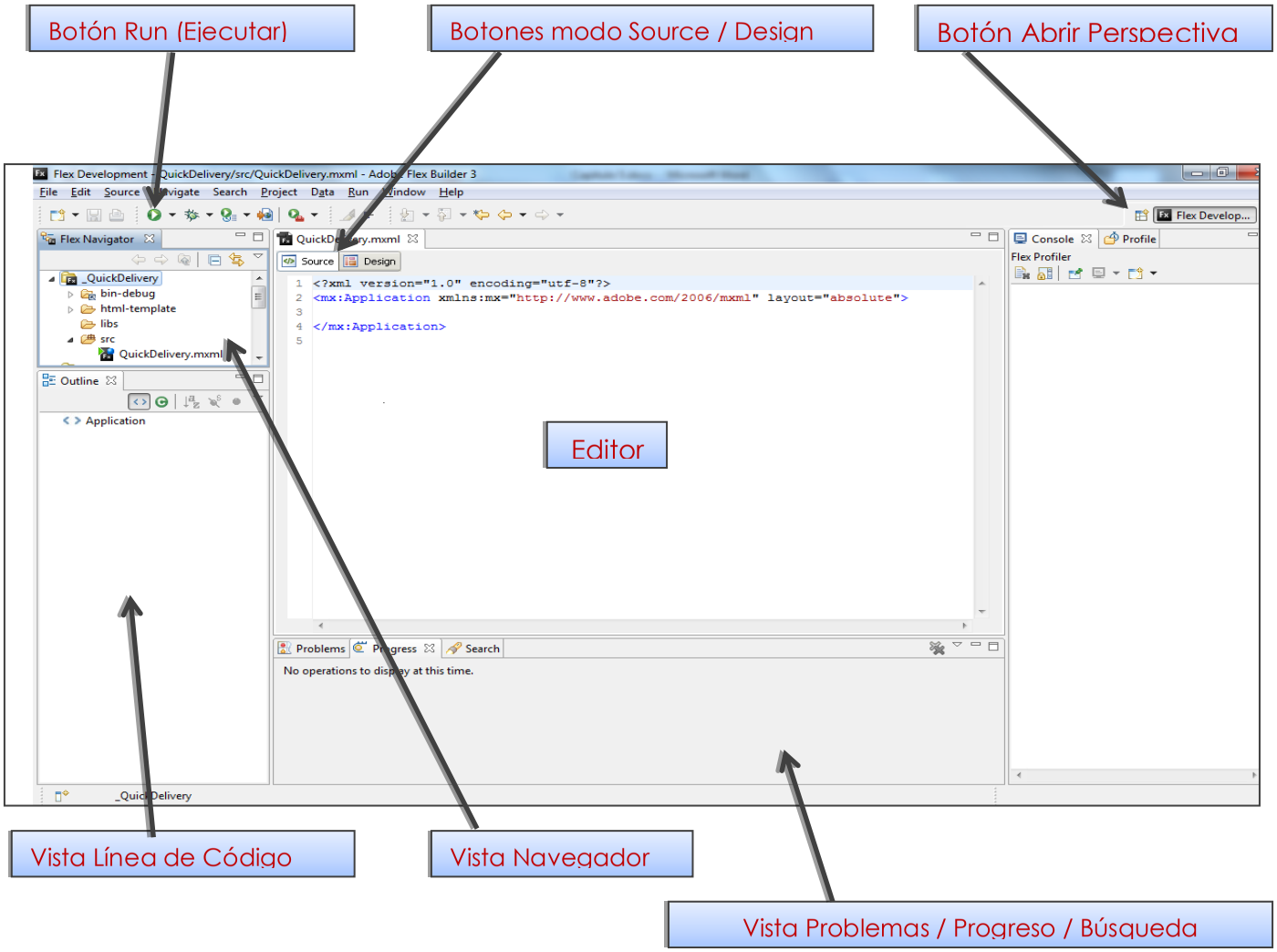


Figura 6. 6 Elementos del entorno de trabajo de Flex Builder 3 modo Source

Antes de continuar es importante conocer cómo se ejecuta una aplicación en Flex y que sucede cuando activamos esta función, además vamos a activar una característica muy importante para realizar la construcción automática de nuestro proyecto y lo haremos en el siguiente menú.

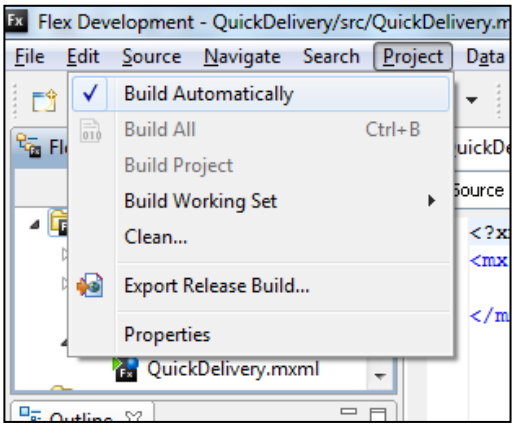


Figura 6. 7 Menú Project para activación de características especiales en Flex Builder 3

Esto con la finalidad de poder ver en tiempo real los ajustes o cambios que realicemos a nuestro proyecto.

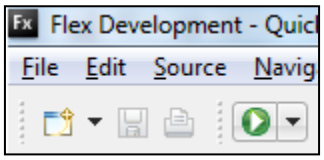


Figura 6. 8 Botón Run para ejecutar aplicaciones en Flex 3

Para ejecutar una aplicación en Flex hacemos clic en el botón Run (tiene la forma de triángulo de color verde), pero ¿qué sucedió exactamente cuándo se apretó este botón? Lo que se ha producido es una gran cantidad de procesos, primero las etiquetas XML del archivo de la aplicación se han transformado a ActionScript. En este

momento ActionScript genera un archivo SWF, que es el formato que entiende Flash Player y este archivo SWF es enviado al navegador.

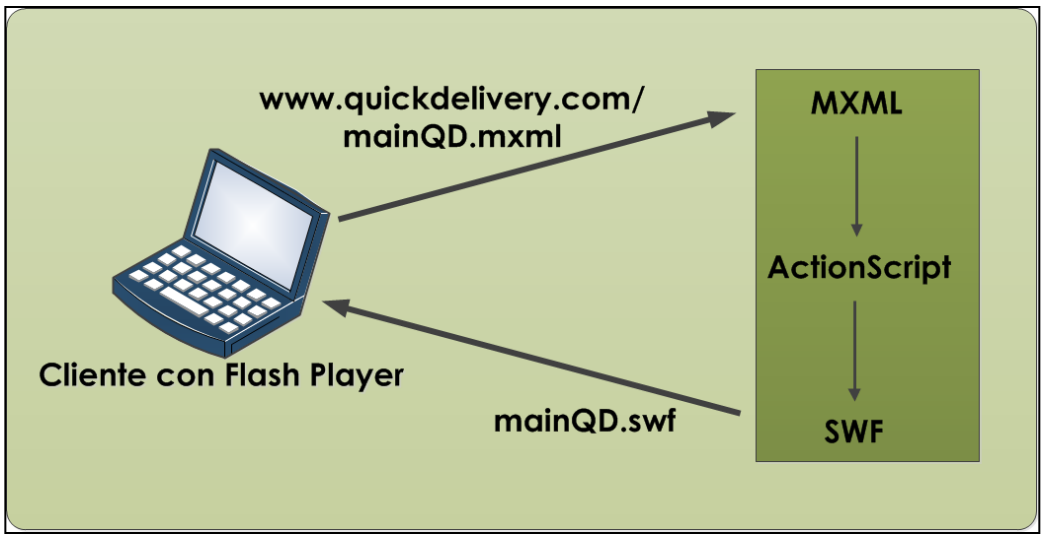


Figura 6. 9 Esquema del proceso de funcionamiento de un sistema RIA

Por el momento hemos presentado como crear aplicaciones en modo Código, ahora vamos a mostrar las características de las que dispone Flex para trabajar en modo Diseño; usando la aplicación QD escogemos en el panel de herramientas "Design" y la perspectiva de Flex Builder cambiará a lo siguiente:

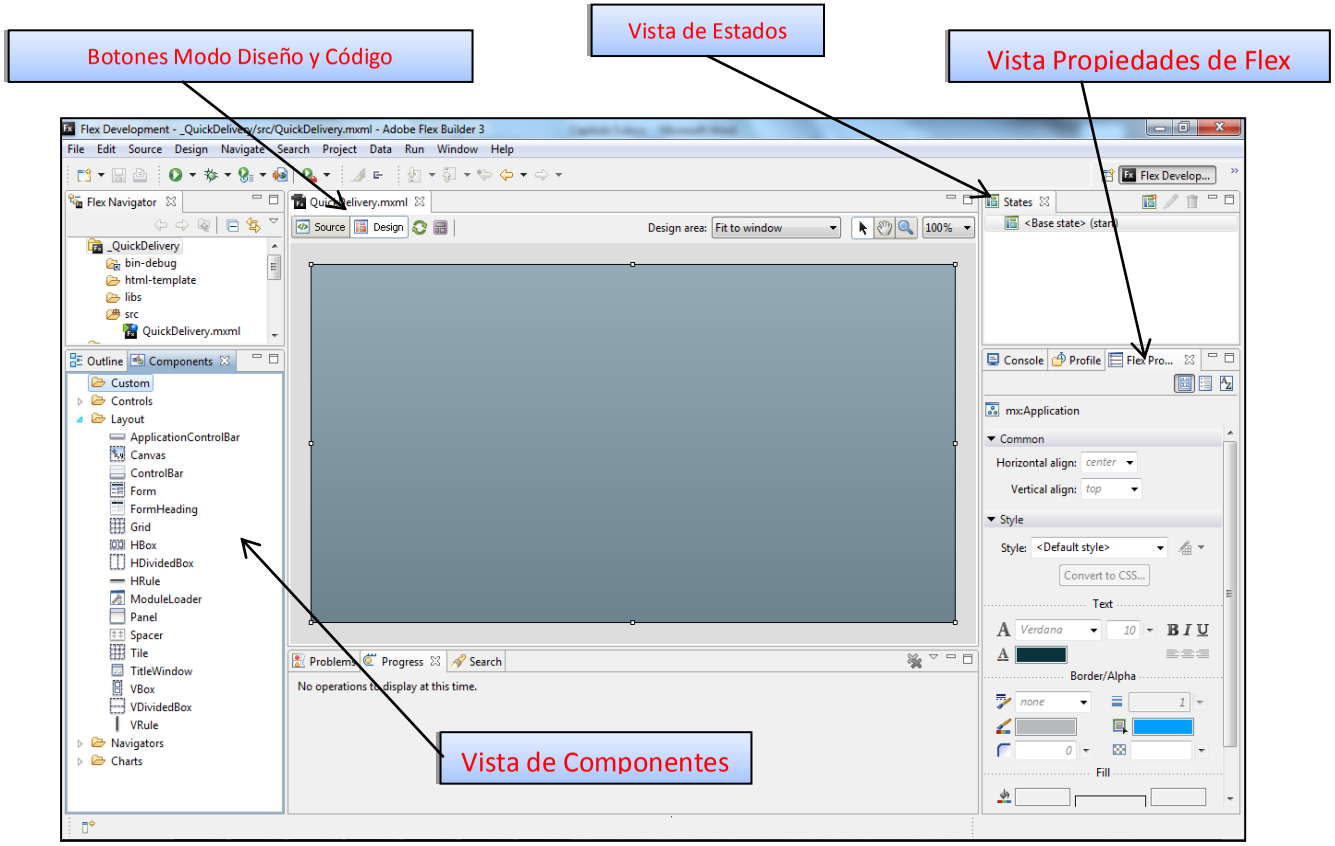


Figura 6. 10 Elementos del entorno de trabajo de Flex Builder 3 modo Design

Ahora que identificamos cada parte de la mesa de trabajo de Flex 3 vamos a explicar cómo interactuar con cada uno de estos elementos, lo primero que tenemos que saber es que podemos usar cualquiera de los componentes localizados en la vista de componentes para arrastrarlos en el espacio de diseño y de esta manera ir armando la parte visual de nuestro proyecto.

Al usar uno de estos componentes y ponerlo en la mesa de diseño podemos observar que en la parte derecha de nuestra pantalla aparece la pestaña Flex Properties, la cual nos detalla las propiedades de este objeto y nos permite cambiarlas tanto lo que respecta a la forma, tamaño, diseño, comportamiento, pudiendo alternar todas estas propiedades en hojas de Estilo o CSS.

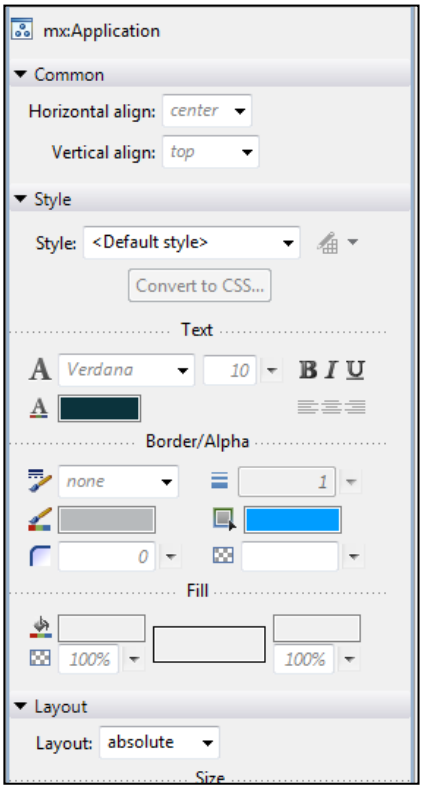


Figura 6. 11 Pantalla Flex Properties, para visualizar las propiedades de los componentes

6.3 Diseño y Codificación del Sistema Quick Delivery

6.3.1 Contenedores

Este es un tema crucial al momento de programar con Flex 3 ya que todo el diseño de Flex está basado en contenedores. Cada contenedor tiene un conjunto de reglas que determinan cómo se diseñan las etiquetas descendientes. Ahora detallaremos las reglas y como se utilizan cada uno de los contenedores.

HBox: Los nodos descendientes se presentan de forma horizontal, cada nodo descendiente aparece a la derecha del nodo descendiente anterior.

VBox: Los nodos descendientes se presentan de forma vertical, cada nodo descendiente se presenta en una situación inferior en la pantalla.

Canvas: Los nodos descendientes aparecen en las coordenadas X e Y especificadas por el desarrollador. Si no se especifica lo contrario, todos los nodos descendientes aparecen en la esquina superior izquierda del contenedor.

Application: Puede configurarse para que se comporte como un contenedor VBOX, HBOX o Canvas mediante el uso del atributo layout.

Title: Los nodos descendientes aparecen en una o más columnas verticales o filas horizontales, comenzando nuevas filas o columnas si es necesario. Todas las celdas del contenedor Title tienen el mismo tamaño.

Panel: Una subclase del contenedor Box, un contenedor Panel puede actuar o bien como un contenedor HBOX, VBOX o Canvas dependiendo del atributo layout especificado (utilizaremos

layout="absolute" para hacer que se comporte como un contenedor Canvas, que es el valor predeterminado).

ControlBar: El contenedor ControlBar se utiliza para acoplar una barra de herramientas a la parte inferior del contenedor Panel o del contenedor TitleWindow. El contenedor ControlBar puede actuar como un contenedor HBOX o VBOX, dependiendo del atributo direction especificada (por defecto es horizontal).

ApplicationControlBar: Puede actuar como un conector HBox o VBox, dependiendo del atributo direction que se ha especificado. El diseño de este contenedor se utiliza para contener componentes que proporcionan acceso a los elementos utilizados a lo largo de la aplicación.

6.3.2 Formatos y Restricciones

Flex proporciona formatos basados en restricciones que nos permiten disponer los elementos de la interfaz de usuario con libertad y precisión, esto en posición absoluta mientras que podemos establecer limitaciones para extender o mover los componentes cuando el usuario cambia el tamaño de la ventana. Hay dos propuestas para los formatos basados en restricciones.

La primera propuesta coloca y da un tamaño a todo en relación con los bordes de un contenedor Canvas y otro contenedor que permita una posición absoluta, como el contenedor Application o Panel.

La segunda propuesta utiliza restricciones mejoradas, en las que las filas y las columnas se configuran en un contenedor que permite un posicionamiento absoluto.

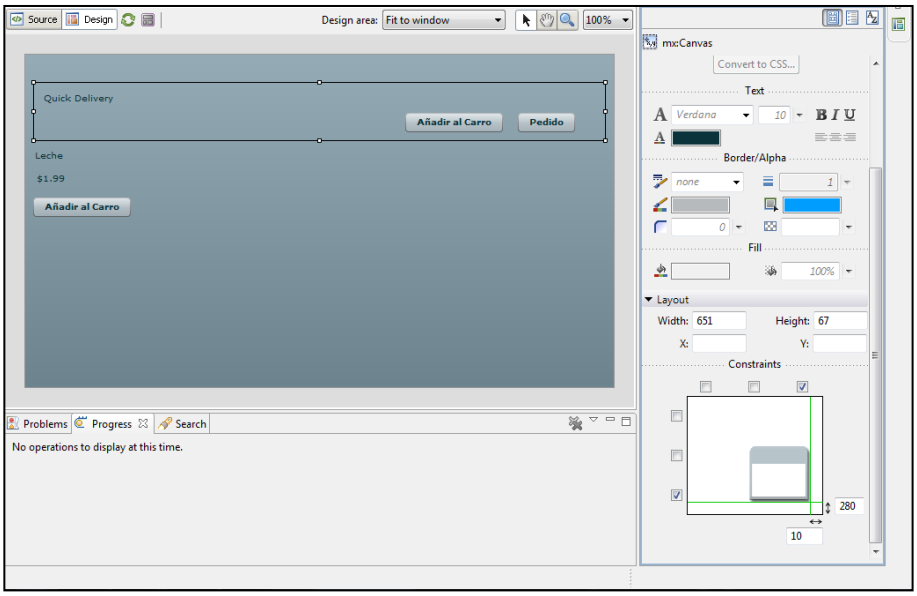


Figura 6. 12 Configuración de contenedores y posicionamiento de componentes

Si queremos mirar el código de la aplicación tendremos algo así:

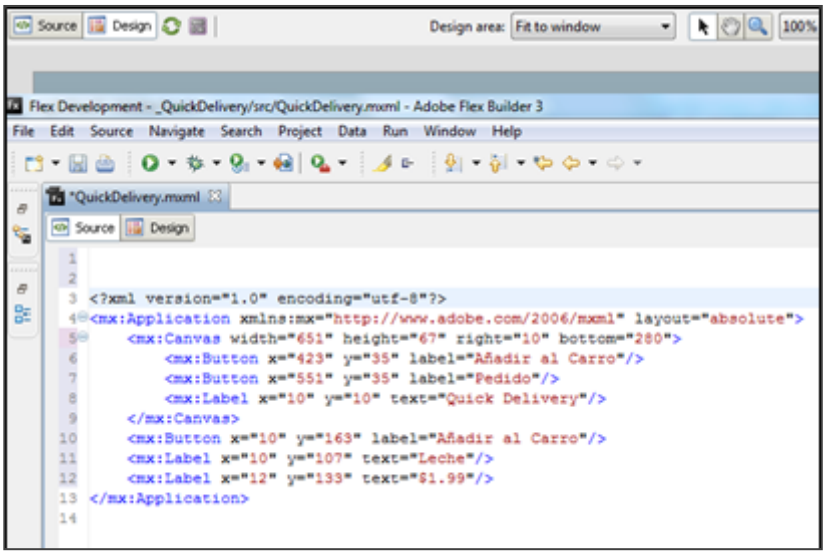


Figura 6. 13 Framewor Flex Modo Source

6.3.3 View States

Una *view state* es una de las muchas vistas que definimos para una aplicación o un componente personalizado. En Flex Builder podemos usar esta característica para crear aplicaciones que cambien su apariencia dependiendo de las tareas que realice el. Por ejemplo, la

aplicación de Quick Delivery comienza mostrando al usuario los distintos productos que se pueden comprar, cuando se empiezan a añadir distintos elementos al carro, queremos añadir algo a la vista para que los usuarios sepan lo que está actualmente en el carro, como el coste total. Finalmente, se necesita una forma para ver y gestionar todos los contenidos del carro de la compra.

Cada página MXML tiene por lo menos un estado, al cual nos referimos como *base view state* (vista de estado básica), que no es más que el diseño predeterminado del archivo. También usaremos *view state* en la vista principal del menú Quick Delivery, para cambiar entre las perspectivas que tenemos para elegir en el Menú de Navegación, alternando entre pantallas como Home, Misión, Visión, Acerca De, Ayuda, Contacto.

Figura 6. 14 Menú de Navegación Aplicación QuickDelivery

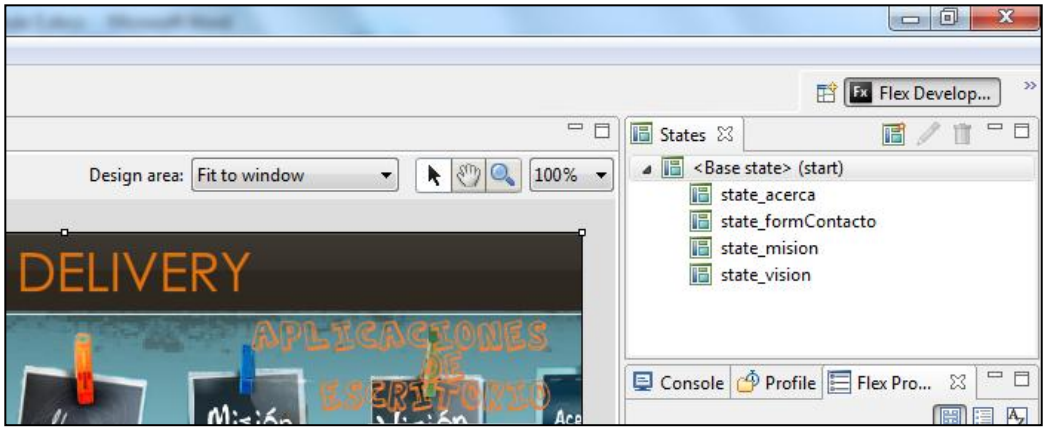
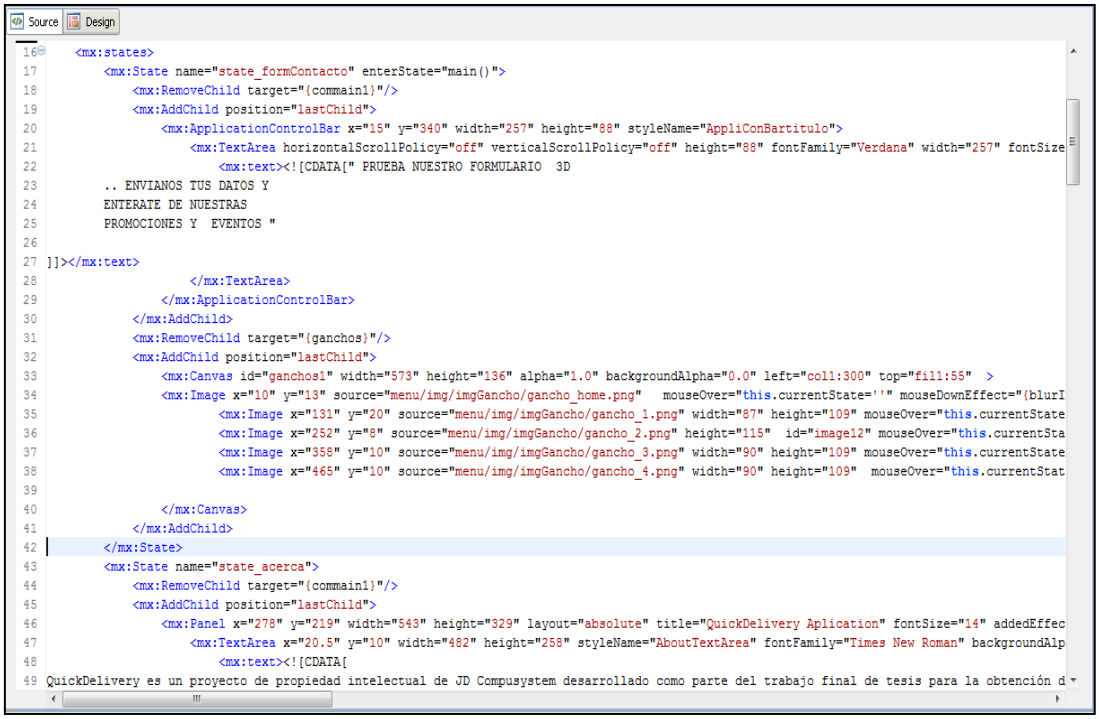


Figura 6. 15 Vista de Estados

El código generado sería el siguiente:



```
Source  Design
16    <mx:states>
17        <mx:State name="state_formContacto" enterState="main()">
18            <mx:RemoveChild target="{commain1}"/>
19            <mx:AddChild position="lastChild">
20                <mx:ApplicationControlBar x="15" y="340" width="257" height="88" styleName="AppliConBartitulo">
21                    <mx:TextArea horizontalScrollPolicy="off" verticalScrollPolicy="off" height="88" fontFamily="Verdana" width="257" fontSize
22                        <mx:text><![CDATA[" PRUEBA NUESTRO FORMULARIO  3D
23        .. ENVIANOS TUS DATOS Y
24        ENTERATE DE NUESTRAS
25        PROMOCIONES Y  EVENTOS "
26
27 ]]></mx:text>
28                    </mx:TextArea>
29                </mx:ApplicationControlBar>
30            </mx:AddChild>
31            <mx:RemoveChild target="{ganchos}"/>
32            <mx:AddChild position="lastChild">
33                <mx:Canvas id="ganchos1" width="573" height="136" alpha="1.0" backgroundAlpha="0.0" left="col1:300" top="fil1:55"  >
34                <mx:Image x="10" y="13" source="menu/img/imgGancho/gancho_home.png"   mouseOver="this.currentState=''" mouseDownEffect="{blurI
35                        <mx:Image x="131" y="20" source="menu/img/imgGancho/gancho_1.png" width="87" height="109" mouseOver="this.currentState
36                        <mx:Image x="252" y="8" source="menu/img/imgGancho/gancho_2.png" height="115"  id="image12" mouseOver="this.currentSta
37                        <mx:Image x="358" y="10" source="menu/img/imgGancho/gancho_3.png" width="90" height="109" mouseOver="this.currentState
38                        <mx:Image x="465" y="10" source="menu/img/imgGancho/gancho_4.png" width="90" height="109"  mouseOver="this.currentStat
39
40                </mx:Canvas>
41            </mx:AddChild>
42        </mx:State>
43        <mx:State name="state_acerca">
44            <mx:RemoveChild target="{commain1}"/>
45            <mx:AddChild position="lastChild">
46                <mx:Panel x="278" y="219" width="543" height="329" layout="absolute" title="QuickDelivery Aplication" fontSize="14" addedEffec
47                    <mx:TextArea x="20.5" y="10" width="482" height="258" styleName="AboutTextArea" fontFamily="Times New Roman" backgroundAlp
48                        <mx:text><![CDATA[
49 QuickDelivery es un proyecto de propiedad intelectual de JD Compusystem desarrollado como parte del trabajo final de tesis para la obtención d
```

Figura 6. 16 Vista del código generado en la creación de Estados

6.3.4 Controles Básicos

Los controles simples son parte de la estructura y ayudan a hacer el desarrollo de las Aplicaciones Ricas de Internet (RÍA) más fácil. Al utilizar controles podemos de forma más sencilla crear la apariencia de nuestra aplicación. Los controles son la base de cualquier RÍA.

Flex incluye una biblioteca de clases considerable tanto para los controles simples como para los complejos. Todas estas clases pueden ser creadas a través de una etiqueta MXML o una clase estándar ActionScript y sus API son accesibles tanto en MXML como en ActionScript. La jerarquía de clases también incluye otras clases que definen el nuevo modelo de evento, además de desplegar atributos que comparten los controles simples.

Situamos los componentes visuales de nuestra aplicación Flex dentro de los contenedores, que proporcionan cuadros delimitadores para texto, controles, imágenes y otros elementos. Todos los controles simples tienen eventos que pueden utilizarse para responder a las acciones de usuario, como hacer clic en un botón, o los eventos del sistema, como diseñar otro componente.

El control Text se utiliza para desplegar múltiples líneas de texto, pero no se puede modificar y no muestra las barras de desplazamiento si la propiedad real ha excedido el espacio disponible. El componente TextArea es útil para visualizar varias líneas de texto, que se puede editar o no, con barras de desplazamiento si el texto disponible excede el espacio disponible en pantalla. Todos los controles de texto admiten HTML 1.0 y una gran variedad de texto y estilos de fuentes.

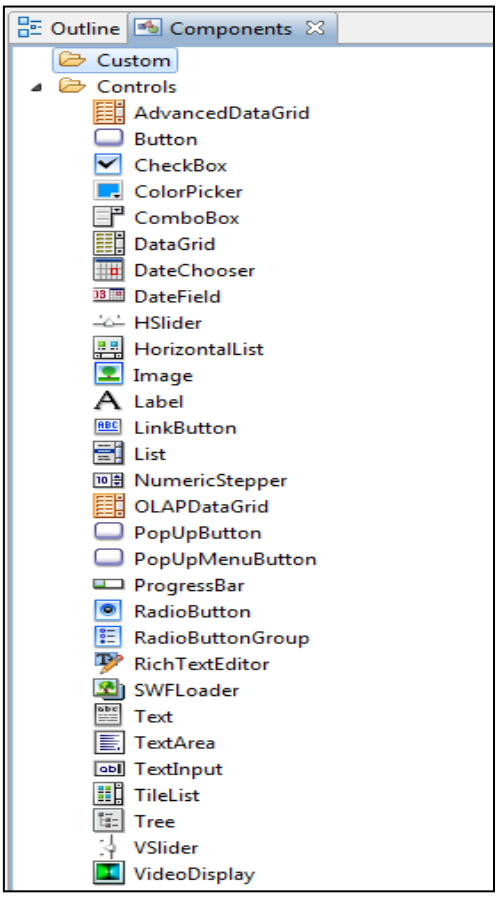


Figura 6. 17 Vista de Componentes de Flex 3

6.3.5 Eventos

Los eventos personalizados se utilizan para construir aplicaciones más fáciles de mantener y que reducen el riesgo de cambio de una parte de la aplicación que forzaría un cambio en otra. Normalmente nos referimos a este concepto como a construcción de una aplicación *loosely coupled* (débilmente acoplada).

Flex utiliza un modelo de programación basado en eventos, lo que quiere decir que estos eventos van a determinar el flujo de la aplicación. Estos eventos son de dos tipos, los eventos del usuario y los eventos del sistema. Los eventos del usuario interactúan por medio de los botones del ratón o del teclado, en cambio los eventos del sistema se usan cuando una aplicación se carga o se cierra, cuando un componente es instanciado o se visualiza además de aquellos componentes que cambian su estado visual como la creación o destrucción de un componente o cuando el componente cambia de tamaño, también hay eventos que ocurren cuando finaliza la carga de datos desde un servidor.

Podemos manejar estos eventos en el código mediante la adición de un controlador de eventos. Los controladores de eventos son las funciones o métodos que se escriben para responder a eventos específicos. También se refiere a veces como detectores de eventos o "event listeners". El modelo de eventos en Flex comprende los sucesos de un objeto y sus subclases, y el envío de modelo de eventos.

Un objeto que requiere información sobre los eventos de otro objeto registra un oyente (listener) con ese objeto. Cuando ocurre un evento, el objeto distribuye el evento a todos los detectores registrados y llama a una función que se solicitó durante el registro. Para recibir los eventos múltiples del mismo objeto, debe registrar su detector para cada evento.

6.3.6 Manejo de Datos Remotos

Para ejemplificar el uso de datos remotos, crearemos un componente dentro de nuestra aplicación llamado "Mantenimiento de Productos", en la cual utilizaremos un control Tree que permite a los usuarios visualizar de forma fácil los detalles de los artículos de la tienda.

En esta aplicación rellenaremos el control utilizando la clase HTTPService para cargar datos XML remotos. Los datos pueden provenir de un servidor remoto o de un archivo externo del mismo servidor, pero en cualquier caso, los datos son remotos a nuestra aplicación; estos datos se transmitirán a través del protocolo HTTP, y permitirá que trabajemos con estos datos XML en diferentes formatos. Por defecto, los datos son analizados en un ArrayCollection, que es una colección de datos con características especiales, y utilizaremos también la funcionalidad de ECMAScript para XML (E4X), que aplica XML como un tipo de datos originales en ActionScript. Para poder utilizar todos estos datos en la práctica, los utilizaremos como código para los controles que tienen estructuras de datos complejas como proveedores de datos. Estos controles, nos permiten visualizar fácilmente conjuntos de datos complejos y permite al usuario navegar por estos conjuntos de datos.

6.3.6.1 Recuperación de datos XML con HTTPService

Es una buena práctica cargar el XML durante el tiempo de ejecución utilizando la clase HTTPService. Para hacer esto de una forma sencilla tratamos de explicar que el componente HTTPService proporciona acceso a los URL y devuelve los datos visualizados en esos URL. Normalmente los datos de los URL tendrán un formato de XML. Se devuelven los datos y puede

seleccionar el formato en el que quiere utilizar los datos devueltos. Para utilizar HTTPService hacemos lo siguiente:

1. Crearemos un objeto HTTPService.
2. Invocamos el método send() del objeto.
3. Usamos los datos devueltos.

6.3.6.2. Creación un objeto HTTPService

Cuando se crea el objeto, hay que especificar el URL que visualizará el servicio y hay que especificar también el evento result, que se transmite cuando los datos son devueltos con éxito por el objeto HTTPService.

```
<mx:HTTPService id="datos"
        url="http://localhost:8300/flexTesis/datos.xml"
        result="resultHandler(event)" />
```

6.3.6.3. Usando los datos Devueltos

Para obtener los datos seguiremos los siguientes pasos

1. El nombre de instancia de HTTPService.
2. La propiedad lastResult.
3. La ruta de la estructura XML para apuntar al nodo repetido de XML.

Ejemplo, recuperaremos datos remotos usando el HTTPService definido como:

```
<mx:HTTPService id="UnitRPC"
        url="http://localhost:8300/flexTesis/unidades.xml"
        result="unitRpc(event)" />
```

El XML a recuperar es el siguiente:

```
<productoUnidades>
    <unidades>
      <unidadNom> Racimo  </unidadNom>
      <unidadIndice> 1  </unidadIndice>
    </unidades>
    <unidades>
      <unidadNom> Docena</unidadNom>
      <unidadIndice> 2  </unidadIndice>
    </unidades>
    <unidades>
      <unidadNom> Unidad  </unidadNom>
      <unidadIndice> 3  </unidadIndice>
    </unidades>
    <unidades>
      <unidadNom> Libra  </unidadNom>
      <unidadIndice> 4  </unidadIndice>
    </unidades>
<productoUnidades>
```

Usaremos lastResult para poder acceder a los datos devueltos, pero es mejor usar la propiedad event.result en el manejador de eventos siempre que se configure con esta información el objeto HTTPService.

El XML devuelto también está disponible en el manejador de eventos. Si tenemos un manejador de resultados de esta forma:

```
private function unitRPCResult(event:ResultEvent):void
{
        units=event.result.productoUnidades.unidades;
}
```

Y con esto puedo acceder a los datos de la unidad XML usando:

Event.result.productoUnidades.unidades.

Los datos devueltos están en event.result, y para acceder al nodo que se repite usamos productoUnidades.unidades.

Para este momento hemos podido acceder al servidor y obtener los datos a partir de un archivo XML para poder trabajar con ellos más adelante. Lo que vamos a hacer es usar un control Tree para visualizar los datos ya obtenidos y presentarlos de forma coherente. Se ha escogido un control Tree para poder navegar más fácilmente por la estructura de datos complejos y rellenar un formulario que vamos a crear.

ActionScript 3.0 contiene un soporte original XML en forma de ECMAScript para XML (E4X).

ECMAScript para XML (E4X) es un lenguaje de programación de extensión que añade XML a ECMAScript (que incluye ActionScript , DMDScript , JavaScript y JScript). El objetivo es proporcionar una alternativa a la interfaz DOM que utiliza una sintaxis más simple para acceder a documentos XML. También ofrece una nueva forma de hacer visible XML. E4X trata a XML como un objeto primitivo (como caracteres, enteros y

booleanos), esto implica un acceso más rápido, mejor soporte, y la aceptación como una estructura de datos de un programa.

Lo que tenemos que hacer para cargar un Tree control con un archivo XML es, usando un HTTPService lo configuramos para que el resultado de la ejecución de este componente sea en formato E4X como lo mostraremos a continuación:

```
<mx:HTTPService id="UnitRPC"
        url="http://localhost:8300/flexTesis/unidades.xml"
        result="unitRpc(event)" resultFormat="e4x"/>

<mx:Model id="prodModel">
    <producto>
        <catID>{int(catID.selectedItem.catID)}</catID>
        <prodName>{prodName.text}</prodName>
        <unilD>{unitID.selectedItem.unitID}</unilD>
        <costo>{Number(cost.text)}</costo>
        <listPrecio>{Number(listPrice.text)}</ listPrecio >
        <descripcion>{description.text}</descripcion>
        <Organico>{Organico.selected}</Organico>
        </bajoGrasa>{</bajoGrasa>.selected}</bajoGrasa>
        <imageName>{imageName.text}</imageName>
   </producto>
</mx:Model>

<mx:Form>
        <mx:FormItem label="Categoria">
            <mx:ComboBox id="catID"
                dataProvider="{cats}"
                labelField="catName"/>
        </mx:FormItem>
```

```
                    <mx:FormItem label="Producto">
                            <mx:TextInput id="prodName"/>
                    </mx:FormItem>
                    <mx:FormItem label="Unidad">
                            <mx:ComboBox id="unitID"
                                    dataProvider="{units}"
                                    labelField="unitName"/>
                    </mx:FormItem>
                    <mx:FormItem label="Costo">
                            <mx:TextInput id="cost" />
                    </mx:FormItem>
                    <mx:FormItem label="Precio">
                            <mx:TextInput id="listPrice" />
                    </mx:FormItem>
                    <mx:FormItem label="Descripcion">
                                    <mx:RichTextEditor id="description"
                                    height="200"/>
                    </mx:FormItem>
                    <mx:FormItem label="Organico">
                            <mx:CheckBox id="isOrganic"/>
                    </mx:FormItem>
                    <mx:FormItem label="Bajo En Grasa">
                            <mx:CheckBox id="isLowFat"/>
                    </mx:FormItem>
                    <mx:FormItem label="Imagen" direction="horizontal">
                            <mx:TextInput id="imageName"/>
<v:SubirArchivo
uploadComplete="handleUploadComplete(event)"/>
                    </mx:FormItem>
                    <mx:FormItem>
<mx:Button label="Añadir Producto" click="doProdAdd()"/>
                    </mx:FormItem>      </mx:Form>
```

El resto de código que mostramos sirve para especificar un modelo de estructura de datos en el primer caso y en la segunda parte creamos el formulario el cual nos servirá para rellenar el resto de información de los productos en la misma que se podrá añadir modificar y eliminar campos de dicho registro.

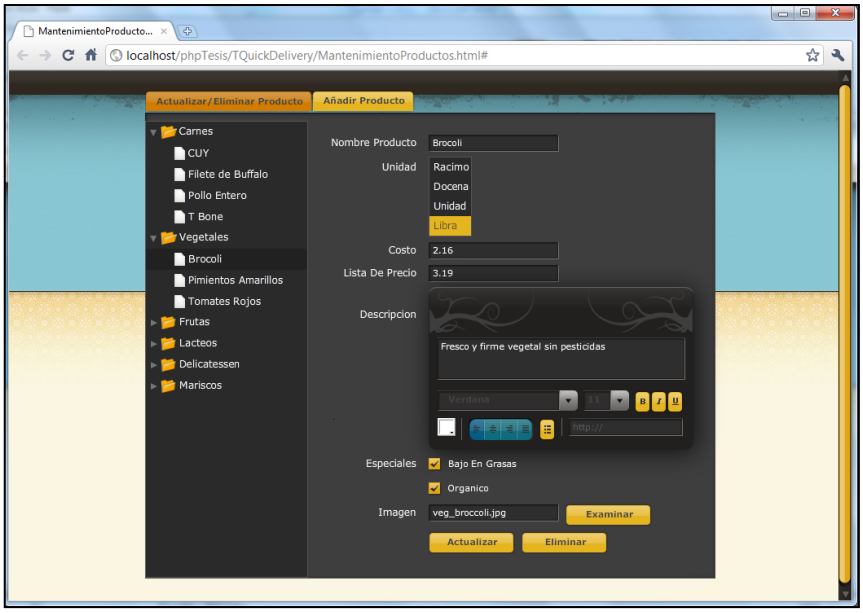


Figura 6. 18 Aplicación QD Mantenimiento de Productos. Ejemplo componentes E4X

6.3.7. Seguridad en la información con Flex Builder 3

Flex está sujeto a las *security sandbox restrictions* de Flash Player, esto significa que una aplicación de un dominio impide cargar datos de otro dominio. Para proporcionar a una aplicación descargada de forma automática desde www.mi-sitio.com el acceso a datos en www.tu-sitio.com, tiene que utilizar un archivo *cross-domain policy* (archivo de política que no contiene ninguna etiqueta). Este archivo, llamado **crossdomain.xml**, especifica qué dominios tienen acceso a los recursos desde Flash Player y está alojado en la raíz del servidor Web al que el archivo SWF al que está llamando. Siguiente a este párrafo ponemos un ejemplo de archivo *cross-domain policy* que permitirá a

cualquier SWF acceder a recursos disponibles en el servidor Web en el que residen:

```
<cross-domain-policy>
<allow-access-from domain="*"/>
</cross-domain-policy>
```

6.3.8. Componentes MXML

Los componentes de Flex propios y todos los componentes que se construyan de carácter personal son clases ActionScript. La clase base para los componentes visuales que hemos estado utilizando y los componentes MXML que construiremos en este capítulo es UIComponent. Esto significa que UIComponent está en lo más alto y todos los demás componentes descienden de este.

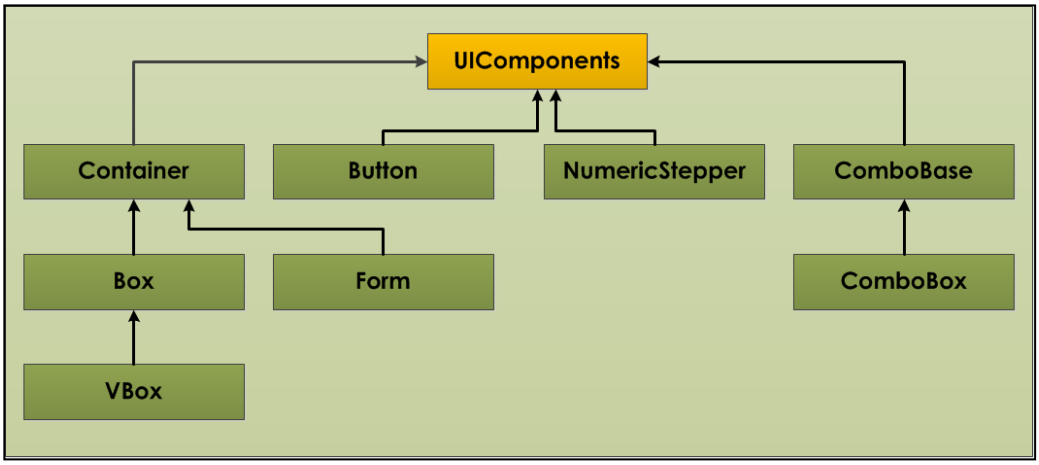


Figura 6. 19 Jerarquía de Componentes UIComponents

Existen conjuntos generales de estas clases dependiendo de su funcionalidad, como clases component, manager y data service. La mayor parte de las clases como component, Application, HBox, y TextInput, todas son clases MXML. También hemos utilizado la etiqueta HTTPService, que no desciende de UIComponent porque es un componente no visual, esta es una etiqueta data service.

Package	mx.rpc.http.mxml
Class	public class HTTPService
Inheritance	HTTPService- HTTPService -AbstractInvoker –EventDispatcher- Object
Implements	IMXMLObject, IMXMLSupport

Figura 6. 20 Descripción Etiqueta HTTPService

6.3.8.1. Creando componentes personalizados

Lo que haremos a continuación será crear un componente personalizado que nos permita ver el detalle de un producto siempre y cuando lo actualicemos o lo eliminemos, algo como un reporte que informe al usuario lo que ha ocurrido con un determinado producto.

Nuestro componente utilizará <mx:List> como su etiqueta raíz en lugar de la habitual <mx:Application>. Los componentes no pueden utilizar esta etiqueta <mx Application> como etiqueta raíz porque esta etiqueta sólo se puede utilizar una vez en cada aplicación.

1. La primera línea que tiene el componente será una declaración de documento XML estándar.

<?xml version="l O" encoding="utf-8"?>

2. Dado que estamos ampliando la funcionalidad de <mx:List> la usaremos como raíz de nuestra etiqueta. Nuestra estructura de componente será como sigue:

<?xml version="l O" encoding="utf-8"?>

<mx:List xmlns:mx="http://www adobe com/2006/mxml">
</mx:List>

3. Una vez configurado el marco de nuestro componente procederemos a llenarlo con la información y los elementos que necesitemos para nuestra pantalla informativa.

Para nuestra necesidad la vista se configurara de la siguiente manera:

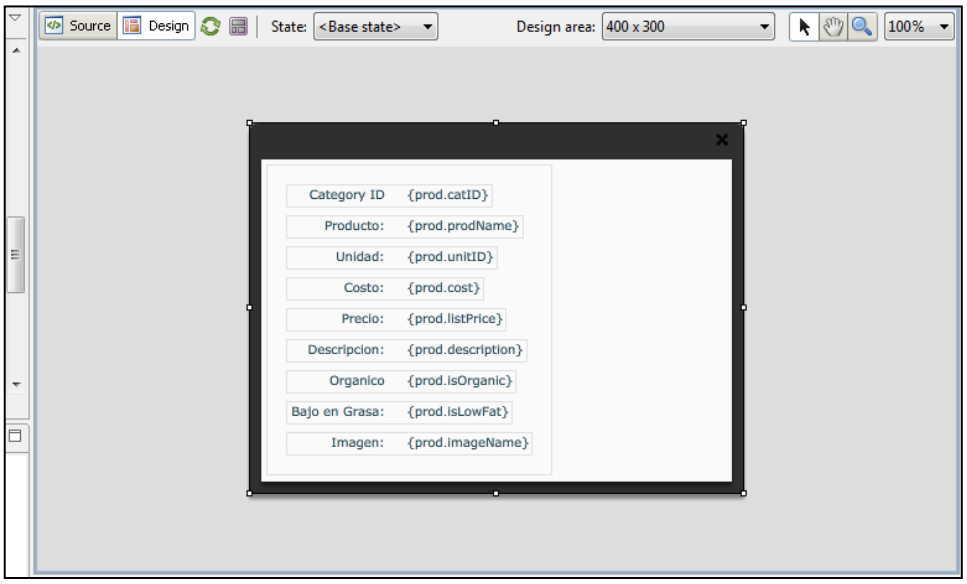


Figura 6. 21 Creación de componente Personalizado para mantenimiento de Productos

Asumimos que se ha creado un archivo con el nombre VentanaConfirmar.mxml en la raíz del proyecto. También se crea el componente en un directorio llamado myComps. Utilizaremos la letra c como prefijo de los componentes de esta carpeta. Por tanto el espacio de nombres XML para añadir a la etiqueta <mx:Application> es xmlns c="myComps.*".

Para terminar crearemos una instancia del componente en el archivo principal de la aplicación:

```
<?xml version="l.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
        xmlns:c="myComps.*">
```

El resultado obtenido con la VentanaConfirmar.mxml sería el que aparece a continuación:

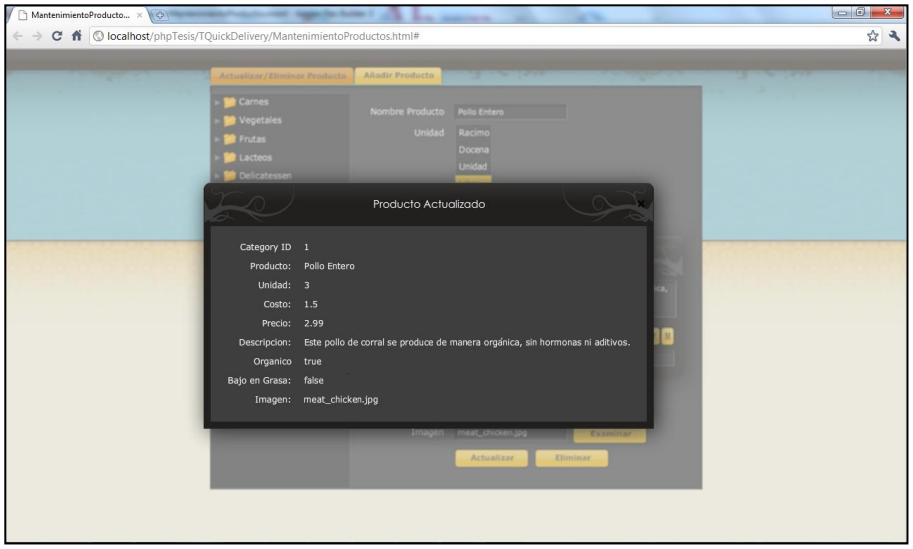


Figura 6. 22 Componente personalizado en funcionamiento con la aplicación Mantenimiento de Productos

Ahora es bueno indicar la importancia de crear componentes personalizados, entra las ventajas que encontramos podemos ver:

- Los componentes hacen que las aplicaciones se construyan, se depuren y se mantengan de forma más fácil.
- Los componentes facilitan el desarrollo de las aplicaciones por parte del equipo.

- Con la planificación adecuada, los componentes nos proporcionarán elementos del código que se pueden reutilizar.

Para facilitar la utilización de los componentes como código reutilizable, debemos hacer que fueran independientes de otros códigos siempre que sea posible. Los componentes deberían operar como piezas independientes de la lógica de la aplicación con una definición clara de lo que les transmitirán los datos y lo que devolverán los datos. El término *loosely coupled* (débilmente acoplado), propio de programación orientada a objetos, se aplica a menudo a este tipo de arquitectura. Además podemos rescatar que trabajar con una arquitectura basada en acoplamiento débil permite usar eventos para notificar a otros componentes los cambios que se han realizado en lugar de exigir a los componentes que conozcan información sobre el resto de la aplicación. Con una arquitectura como ésta, los componentes pueden ser reutilizados en multitud de aplicaciones sin exigir una estructura concreta a las aplicaciones.

Como anteriormente hemos indicado en el capítulo 4, en estas aplicaciones pondremos en práctica la arquitectura MVC.

6.3.9. Flujo y la propagación de eventos

Es importante conocer como maneja los eventos Flash Player; si el evento de destino no es un elemento visible en pantalla, Flash Player puede despachar el objeto evento directamente hasta el destino deseado. Sin embargo, si el destino es un elemento visible en pantalla, Flash Player despacha el evento y viaja desde el contenedor más exterior (el contenedor Application), a través del componente de destino y luego vuelve al contenedor Application.

Event flow o flujo de eventos es una descripción de cómo ese objeto evento viaja a través de una aplicación. Como hemos visto hasta ahora, las aplicaciones Flex se estructuran en una jerarquía nodo ascendiente nodo descendiente, con el contenedor Application como el nodo ascendiente de primer nivel. Como nos hemos dado cuenta flash.events.EventDispatcher es la superclase para todos los componentes en Flex, esto significa que cada objeto en Flex puede utilizar eventos y participar en el flujo de eventos, todos pueden escuchar eventos con el método addEventListener() pero escucharan el evento solo si el objeto que escucha es parte del flujo de eventos.

Cuando se produce un evento, el objeto event hace un viaje completo desde la raíz de la aplicación a través de cada contenedor en su camino hasta el componente que es responsable del evento, lo que se conoce como el target (destino del evento).

El flujo de eventos se divide en tres partes:

- **Capture phase (Fase de captura):** Comprende todos los contenedores desde la aplicación base hasta el contenedor que tiene el evento de destino.
- **Target phase (Fase de destino):** Está formado únicamente por el nodo destino.
- **Bubbling phase (Fase de propagación):** Comprende todos los elementos que se encuentran en el viaje de vuelta desde el destino hasta la aplicación raíz.

La figura 6.23 describe una rama de una aplicación en la que un Button está contenido dentro de un HBox, que a su vez está contenido por un Panel, que está en la raíz de la Aplicación.

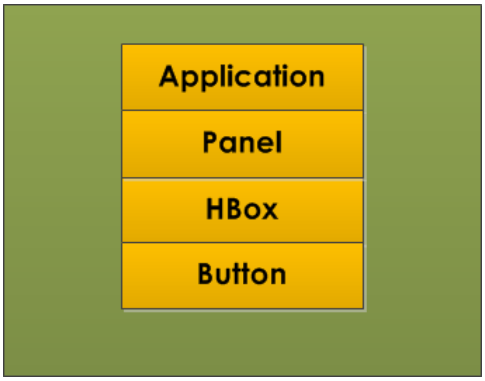


Figura 6. 23 Rama de una Aplicación

Si un usuario hace clic en Button, Flash Player despacha un objeto event al flujo de eventos. El viaje del objeto comienza en Application, se desplaza hacia abajo hasta Panel, se mueve hasta HBox y llega, finalmente, a Button. El objeto event entonces bubbles (se propaga) a través de HBox y Panel en su viaje de vuelta.

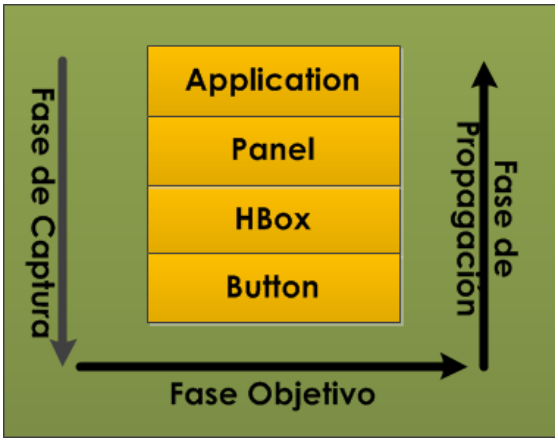


Figura 6. 24 Flujo de Eventos

6.3.10. Programación del Carro de Compra

Usando el control DataGrid crearemos nuestro carro de compras, en el podremos manejar los ítems a comprar. DataGrid nos permite especificar las columnas de forma explícita a través de DataGridColumn. Esto se lleva a cabo mediante el siguiente código:

```
<mx: DataGrid >
    <mx:columns>
        <mx: DataGridColumn dataField=" ">
        <mx: DataGridColumn>
        <mx: DataGridColumn>
    </mx:columns>
</mx:DataGrid>
```

El atributo datafield se utiliza para definir la columna en el conjunto de datos de una columna determinada. El orden en el que aparecen DataGridColumns es el orden en el que aparecen las columnas de izquierda a derecha en DataGrid. Cada DataGridColumn admite un gran número de atributos que afectan al funcionamiento de DataGrid y a su interacción con la columna.

1. Creamos la aplicación carro.mxml desde el directorio **quickDelivery/src/views/QuickDelivery**.

2. Usando la etiqueta <mx: DataGrid> Configuramos Width (Ancho) y Height (Alto) al 100 por 100, draggableColumns a false y editable a true.

```
<mx:DataGrid
        id="cartView"
        dataProvider="{cart.altems}" width="100%" height="100%"
        editable="true" draggableColumns="false">
        <mx:columns>
        </mx:columns>
</mx:DataGrid>
```

Estamos especificando editable como true porque permitiremos que una de las columnas sea modificada por el usuario en unos

cuantos pasos. El DataGrid permanece vinculado a los mismos datos que hemos establecido con anterioridad (cart aItems). También tenemos que establecer el atributo draggableColumns a false porque el valor predeterminado es true, y no queremos que las columnas se puedan mover.

3. Definimos una etiqueta <mx:DataGr idColumn> para el nombre del producto y la colocamos en la parte superior de la lista de columnas.

Configuramos dataField a producto, editable a false y headerText a Producto.

```
<mx:DataGridColumn dataField="producto"
headerText="Producto" editable="false" />
```

El atributo headerText especificará el texto del encabezamiento de DataGridColumn. Si no lo especificamos cogerá el valor del atributo dataField.

4. Definimos <DataGridColumn> para visualizar la cantidad y situarla después del último <mx:DataGridColumn>. Esta columna se utilizará para permitir a los usuarios cambiar la cantidad del producto que quieran comprar.

5. Definimos <mx:DataGridColumn> para visualizar los subtotales de cada elemento y colocamos luego del último <mx:DataGridColumn>.

Configuramos dataField a subtotal, editable a false y headerText a Cantidad.

```
<mx:DataGridColumn dataField="Subtotal"
headerText="Cantidad" editable="false" />
```

La columna presentará el coste de un producto en concreto.

Guardamos y ejecutamos la aplicación y añadimos el producto el producto Bife Asado al carro de la compra y hacemos clic en Ver Carrito. Además hemos añadido los botones de eliminar y el link de Continuar comprando.

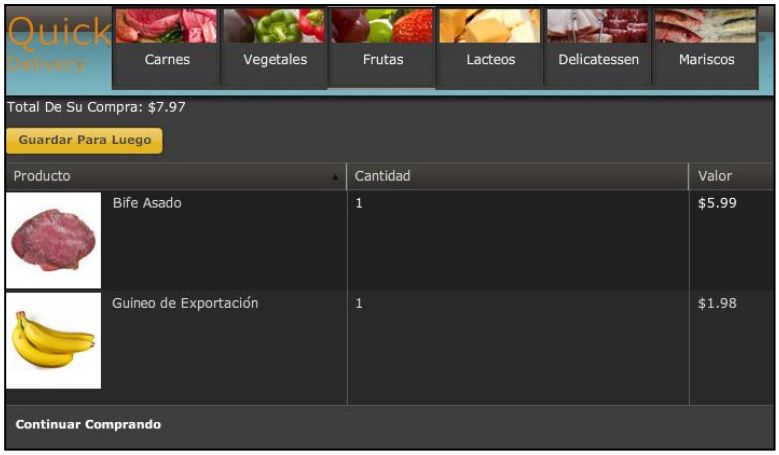


Figura 6. 25 Carro de compras

6.3.11. Control de edición en la línea de DataGridColumn

En un DataGrid tenemos la posibilidad de especificar que una columna de datos presentada pueda ser modificada por el usuario cuando sea la celda la que reciba la entrada de información. Esto se hace configurando el atributo editable a true. El control de edición predeterminado para la columna es un campo de texto. Es posible especificar al editor que queremos utilizar cuando manejamos los datos a través de los atributos itemEditor y editorDataField, que especifican el atributo del control editor utilizado para cambiar el valor para la celda y qué atributo en ese control debería considerar el conjunto de datos para conseguir el valor modificado.

Ponemos a continuación los controles predeterminados que podemos especificar:

- Button
- CheckBox
- ComboBox

- DateField
- Image
- Label
- Text
- TextArea
- TextInput

Usando el DataGrid que tenemos en el archivo QuickDelivery.mxml configuramos itemEditor como NumericStepper, editorDataField como value y editable a true:

```
<mx:DataGridColumn dataField="cantidad"
        itemEditor="mx.controls.NumericStepper"
        editorDataField = "value" editable="true"
        headerText="Cantidad" />
```

Con estos pasos tendremos configurado nuestro carro de compras para poder editar la información del campo cantidad y así poder cambiarla de forma más interactiva.

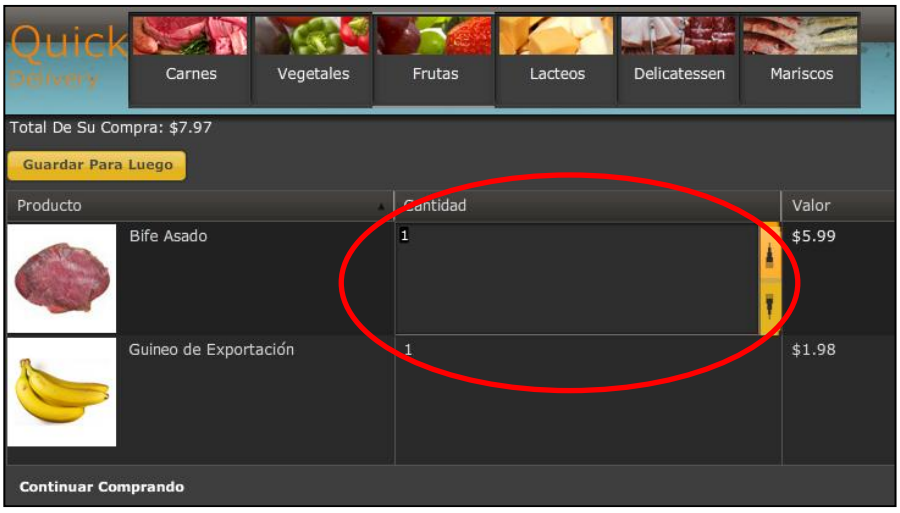


Figura 6. 26 Ejemplo Carro de Compras con función de control de Edición en Data Grid

6.3.12. Uso de la función Arrastrar y Soltar (DRAG AND DROP) para añadir productos al carro de compras

Drag and Drop (arrastrar y soltar) es una técnica de la interfaz de usuario muy común en las aplicaciones de los ordenadores de escritorio. Por otra parte, no era tan común en las aplicaciones Web hasta que se desarrollaron las Aplicaciones Ricas de Internet (RÍA).

Para poner en funcionamiento la característica de arrastrar y soltar en una aplicación Flex tendremos que utilizar Drag and Drop Manager y las herramientas que éste habilita. Drag and Drop Manager nos permite escribir una aplicación Flex en la que los usuarios pueden seleccionar un objeto, arrastrarlo por encima de otro y soltarlo en el segundo objeto.

Para explicar esta funcionalidad, en nuestra aplicación de comercio electrónico, queremos que un usuario pueda hacer clic en un producto, arrastrarlo al carro de la compra y soltarlo para añadirlo al carro de la compra.

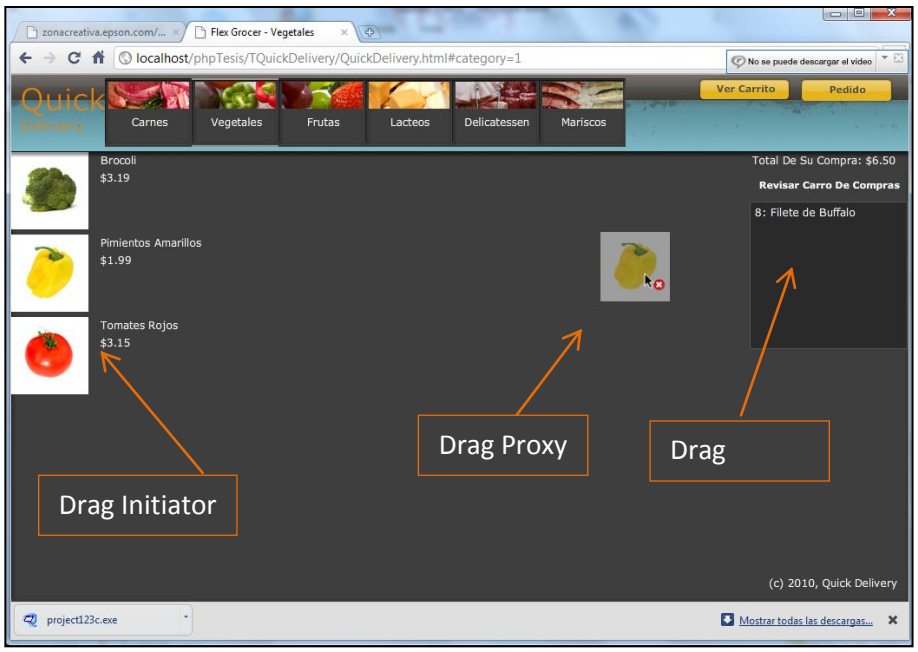


Figura 6. 27 Función Drag and Drop Carro de compras QuickDelivery

6.3.12.1 Drag and Drop Manager

Terminología del componente D & D Manager:

Termino	Definición
Drag initiator	Componente o elemento de un componente desde donde se inicia el proceso arrastrar
Drag source	Datos que se arrastran
Format	Propiedad de DragSource que permite que un objeto sea arrastrado o no hacia otro objeto Los datos que están en DragSource también se encuentran asociados con el formato; el tipo de datos del formato son cadenas simples
Drag proxy	Imagen visualizada durante el proceso de arrastrar
Drop target	Componente sobre el que está el *drag proxy*

Tabla 4 Componentes Drag & Drop

Existen tres fases en una operación para arrastrar y soltar:

1. **Initiating (Inicio):** Un usuario hace clic en un componente Flex o bien en un elemento de un componente Flex y empieza a mover el componente o elemento mientras se mantiene pulsado el botón del ratón. El componente o elemento es el *drag initiator*.

2. ***Dragging*** (Arrastrar): Mientras se mantiene pulsado el botón del ratón, el usuario mueve el ratón por la pantalla. Flex presenta una imagen llamada drag proxy y el objeto no visual asociado llamado drag source contiene los

datos asociados con el componente o elemento que se está arrastrando.

3. ***Dropping*** (Soltar): Cuando el usuario se mueve sobre otro componente que lo permita, el elemento puede soltarse sobre el drop target. Los datos se insertan entonces de alguna forma en el nuevo componente.

Los componentes de Flex pueden dividirse en dos grupos en cuanto a la admisión de la característica de arrastrar y soltar, aquellos con una funcionalidad para arrastrar y soltar mejorada y aquellos que no la tienen. A continuación aparecen los controles basados en listas que tienen un soporte mejorado para arrastrar y soltar:

- DataGrid
- PrintDataGrid
- Tree
- Menú
- List
- HorizontalList
- TileList

Esto significa que será más fácil para nosotros los desarrolladores utilizar la característica de arrastrar y soltar con esos controles que tienen un soporte mejorado. En muchos casos se necesitará únicamente establecer una única propiedad *value* para cada uno de los controles involucrados en la operación de arrastrar y soltar.

Es importante mostrar los eventos de *drag initiator* y de *drop target para trabajar de forma correcta:*

Eventos Drag	Descripción
mouseDown y mouseMove	Aunque no son eventos drag, estas clases de eventos MouseEvent se usan para empezar un proceso de arrastrar y soltar cuando no se usan componentes dragEnabled. El evento mouseDown se transmite cuando el usuario selecciona un control con el ratón y mantiene pulsado el botón del ratón. El evento mouseMove se transmite cuando se mueve el ratón.
dragComplete	Se transmite cuando se completa una operación arrastrar, bien (clase DragEvent) cuando los datos arrastrados se sueltan en un drop target o cuando la operación arrastrar y soltar finaliza sin haber realizado la operación soltar.
dragEnter	Se transmite cuando un drag proxy se mueve sobre el destino desde fuera del destino
dragOver	Se transmite cuando el usuario mueve el ratón sobre el destino después de un evento dragEvent.
dragDrop	Se transmite al soltar el botón del ratón sobre el destino.
dragExit	Se transmite cuando el usuario arrastra desde fuera del drop target, pero no suelta los datos en el destino.

Tabla 5 Eventos Drag

Continuando con el desarrollo de nuestro sistema implementaremos este elemento de arrastrar y soltar para lograr que interactúen tanto el DataGrid de productos con el List del carro de compras como mostramos en la figura 6.27.

Lo primero que haremos es añadir un escuchador de eventos dragDrop en List y llamaremos a la función con el nombre **doDragDrop()** en el ActionScript ejecutado cuando se transmite el evento. Transmitimos el objeto event como un parámetro de la función.

```
<mx:List id="targetList"
        width="200"
        dropEnabled="true"
        dataProvider="{targetListDP}"
        dragDrop="doDragDrop(event)"/>
```

En el bloque del Script crearemos una función privada con el nombre **doDragDrop()** con datos de tipo void como lo indicamos en el código anterior. La función tiene que aceptar un parámetro llamado event con datos de tipo DragEvent. También tenemos que importar la clase:

```
Import mx.events.DragEvent;
```

Se llamará a esta función cuando el usuario suelte *drag proxy* sobre List, que es el *drop target* en esta aplicación.
Como primera línea del código de la función crearemos una variable local a la función llamada **dgRow** con datos de tipo Object y la configuramos igual al nuevo Object.

```
var dgRow :0bject=new Object();
```

Esta variable se utilizará para almacenar información sobre la fila arrastrada de DataGrid y que está almacenada en el objeto DataSource.

Como segunda línea del código de la función configuramos la variable dgRow igual a los datos en el objeto DragSource asociado con el formato Ítems. Utilizamos el método dataForFormat().

```
dgRow=event.dragSource.dataForFormat("Ítems");
```

El método dataForFormat () es un método de la clase DragSource. Recupera desde el objeto DragSource los datos asociados con el objeto, en este caso, Ítems.
A continuación añadimos el nombre del producto a List utilizando el método addItem() del dataProvider de List.

```
argetList.dataProvider.addItem(dgRow[0].ñame);
```

Ejecutamos la aplicación y arrastramos un ítem desde DataGrid hasta List.
Como lo último de esta parte usamos la clase del método preventDefault() para cancelar el comportamiento predeterminado del evento.

```
event.preventDefault() ;
```

En este caso cancelamos el comportamiento predeterminado.
Nota: Todos los eventos pueden cancelarse, al cancelar este evento evitamos que aparezca la línea [object Object] en List.

6.3.13 Función Save For Later (Guardar para Luego)

Ahora en nuestro carro de compras implementaremos la función de "Guardar para Luego". Esto nos permitirá tener la capacidad de recordar información sobre un usuario específico como preferencias, búsquedas, etc., dándonos la capacidad de conservar la información.

La conservación de los datos se puede lograr en el servidor mediante la asociación de un usuario con una identidad de acceso y luego transmitiendo la información específica de vuelta al servidor, esta información se almacena en una base de datos y puede ser descargada nuevamente en la aplicación cuando sea necesario. Ahora mediante Flex 3 podemos guardar esta información en el lado del cliente haciendo uso de Flash Player gracias a una clase con el nombre de SharedObjects. Estos Shared Objects o también llamados Objetos Compartidos trabajan como las cookies de HTTP con la diferencia de que son más potentes ya que nos permiten guardar estructuras de datos mucho más complejos.

Para poder trabajar con los Objetos Compartidos tendremos que manipular sus funciones mediante la utilización de código ActionScript, los datos asignados a los Objetos Compartidos se almacenan en el momento en que se elimina el archivo SWF, esto es al salir del navegador o cuando se cambia a una página Web diferente o como es nuestro caso cuando se produce un evento.

Los Objetos Compartidos tienen las siguientes características:

- ✓ Se almacenan en el ordenador del usuario final.
- ✓ Pueden albergar estructuras complejas de Datos.
- ✓ Tienen la extensión **.sol**.
- ✓ No contienen ni métodos ni funciones.
- ✓ Por defecto su tamaño máximo es de 100KB. Este valor puede ser modificado.

- ✓ Como las cookies los objetos compartidos no pueden leerse desde dominios diferentes. Flash Player puede leer estos archivos siempre que hayan sido creados en el mismo dominio que el archivo SWF original.

Implementaremos la clase ShareObject para almacenar datos en el carro de compras.

Vamos a añadir un botón, para que cuando se haga clic en este se haga la lectura del carro de compras y escriba esta información en el objeto compartido del computador del cliente. Esta característica permitirá a los usuarios acceder al carro de compras en cualquier momento antes de que se complete la compra.

Para esto crearemos un nuevo botón en la aplicación CarroCompra.mxml, con el nombre de guardar para luego.

```
<mx:Button label="Guardar Para Luego"
click="carro.guardarCarro()"/>
```

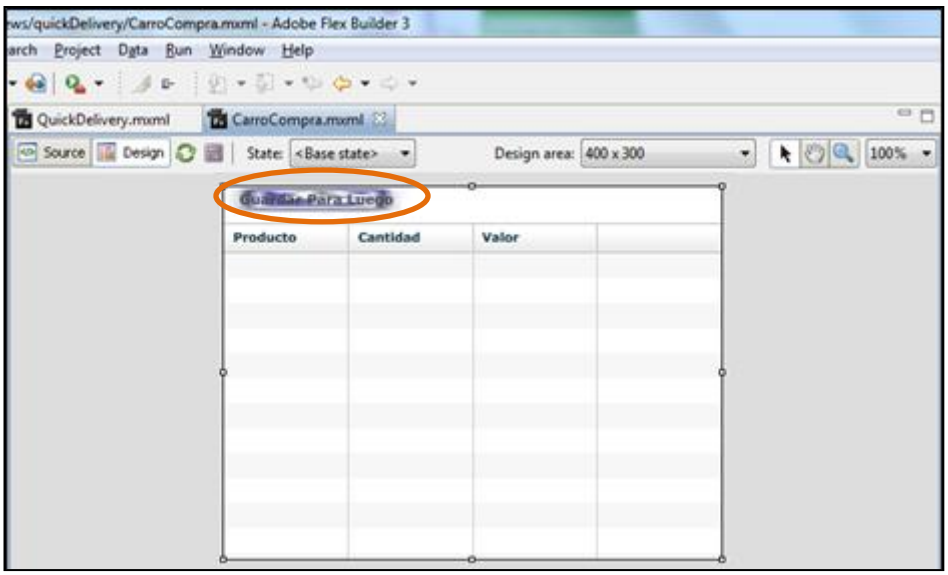


Figura 6. 28 Creación de Botón Guardar Para Luego

Ahora crearemos la función guardarCarro, la cual se encontrará en la carpeta valueObjects en el archivo carroCompra.as, y añadiremos el siguiente código:

```
import flash.net.SharedObject;
```

Con esta línea invocamos a las librerías necesarias para poder utilizar los SharedObject de Flash Player.

Luego crearemos la función guardarCarro con el método estático getLocal() de la clase SharedObject para declarar un nuevo objeto compartido con el nombre de soCarro, y transmitimos el parámetro carroInfo al método getLocal().

```
public function guardarCarro():void{
        this.soCarro = SharedObject.getLocal("carroInfo");
        this.soCarro.data.aCarro = new Array();

        var len:int = aItems.length;
        for (var i:int = 0;i < len;i++){
                this.soCarro.data.aCarro[i] = this.aItems.getItemAt(i);
                        } }
```

Esto creará un nuevo Objeto Compartido y tambien un archivo en el computador del cliente con el nombre de carroInfo.sol.
También con el método guardarCarro() se declara un nuevo array con el nombre de aCarroen la propiedad data de SharedObject.
El bucle For buscará en el array dentro de SharedObject y subtituirá los valores de este cada vez que se presione el botón guardar para luego.

Una vez creado nuestra función de guardar para luego vamos a ver como cargaremos estos datos cuando el cliente decida continuar con

su operación. Para esto necesitamos cargar los datos desde el carro archivo carroinfo.sol hacia el DataGrid del Carro de compras.
Aquí es donde usaremos el archivo carro.mxml y en la cabecera dentro de la etiqueta <mx:WindowApplication> pondremos el escuchador del evento creationcomplete y lo apuntaremos al método cargarCarro(), esto lo haremos para asegurarnos que el DataGrid se encuentra disponible.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx=http://www.adobe.com/2006/mxml
creationComplete="carro.cargarCarro()">
```

Ahora iremos al archivo carroCompras.as y crearemos la siguiente función:

```
public function cargarCarro():void{


        this.soCarro = SharedObject.getLocal("carroInfo");
        if ( this.soCarro.data.aCarro != undefined ){
                var len:int = this.soCarro.data.aCarro.length;
                for (var i:int=0;i<len;i++){
                        var miProducto:Producto =
                Producto.buildProduct(this.soCarro.data.aCarro[i].pr
                oduct);
                        var miCantidad:int =
                        this.soCarro.data.aCarro[i].cantidad;
                        var miItem:CarroCompraProd = new
                        CarroCompraProd(miProducto, miCantidad);
                        this.addItem(miItem);
                }
        } }
```

En esta función construimos el ArrayCollection desde la estructura de los datos almacenados en el Objeto Compartido, con el método estático getLoca() de la clase SharedObject leemos el Objeto Compartido creado anteriormente y lo almacenamos en la variable soCarro. Ahora nos aseguramos que la variable aCarro sea reconocida desde un principio, ya que esta es reconocida desde el momento que empezamos a utilizar los datos y nos dará un error antes de este evento, por lo que tenemos que añadir la lógica condicional que nos asegura que esta variable no sea "Undefined".

El bucle buscará a través del array aCarro almacenado en el Objeto Compartido, y pondrá esta información en el ArrayCollection aItems. El método buildProduct() construirá un nuevo objeto value Producto basado en la clase antes escrita Producto. Para construir el ArrayCollection aItems utilizaremos el método addItem() de carroCompras. Este método comprueba si el elemento ya está incluido en el carro, gestiona la cantidad de cada elemento y actualiza los subtotales de cada elemento. Añadimos el método limpiaCarro(), el cual contendrá la línea aItems.removeAll(), con la que podremos limpiar el contenido del objeto compartido soCarro.

Al ejecutar esta aplicación nos daremos cuenta que nuestro botón guardar para luego estará funcionando correctamente.

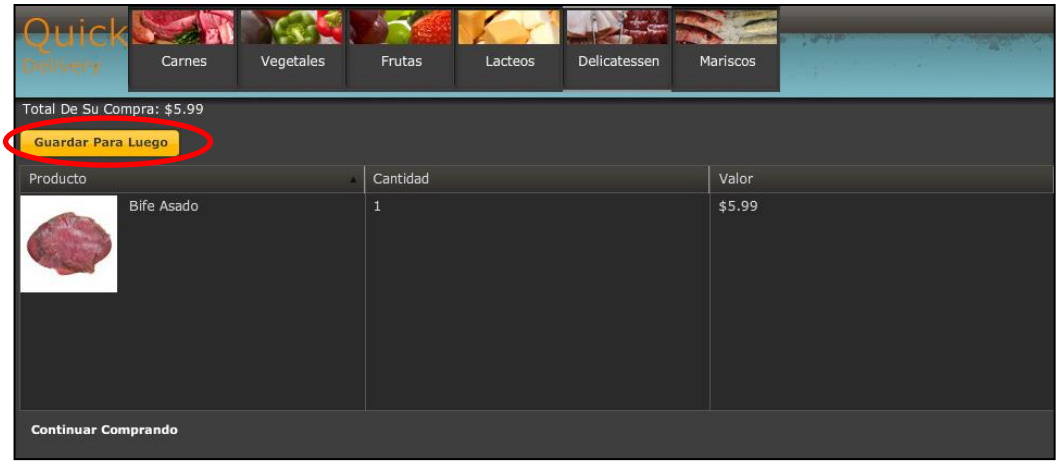


Figura 6. 29 Puesta en funcionamiento botón Guardar para luego

Figura 6. 30 Archivo creado por los Objetos compartidos

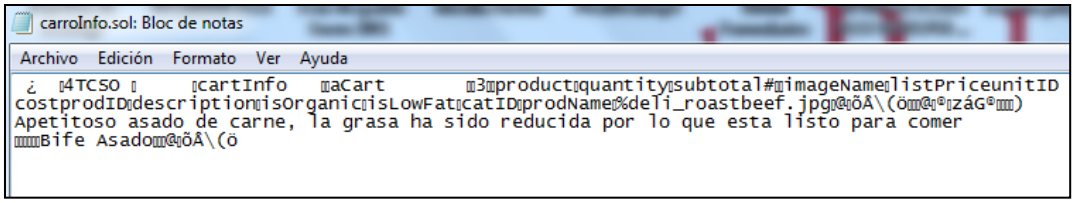


Figura 6. 31 Detalle del contenido del SharedObject carroInfo.sol

Como vemos en el contenido de este archivo se encuentra el producto que seleccionamos en nuestro carro de compras antes de pulsar el botón guardar para luego.

6.3.14 Navegación en QuickDelivery

Lo que pretendemos en esta parte de nuestra tesis es implementar la posibilidad de navegación para darle al usuario la capacidad de moverse por toda la aplicación sin necesidad de generar recarga en la página.

Con Adobe Flex 3 podemos implementar la navegación utilizando un conjunto de contenedores conocidos como contenedores de navegación, los mismos que controlan los movimientos del usuario mediante un grupo de contenedores descendientes (HBox, VBox, Canvas, contenedores de navegación etc.).

Muchas de las funciones como hacer clic en un botón y moverse desde la página principal o el proceso de verificación son realizadas por el usuario, otros en cambio, pueden ser controlados por nosotros

como el proceso de verificación de tarjeta de crédito, usuario, etc. En los que el usuario no puede pasar a la pantalla siguiente si no se valida correctamente.

En la figura 6.32 mostraremos los contenedores navigator más comunes.

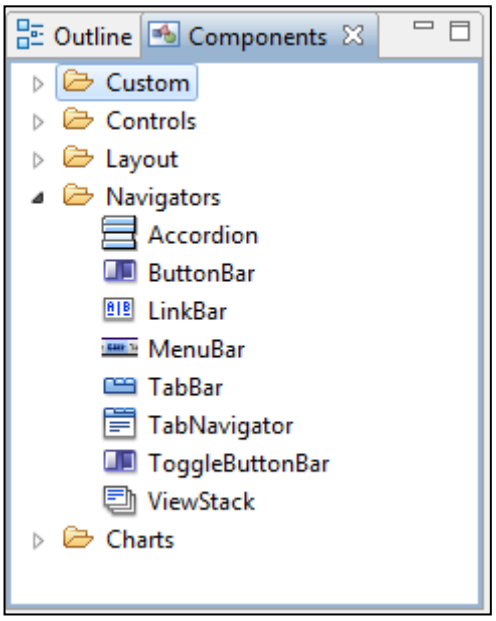


Figura 6. 32 Componentes Navigator

Uno de los elementos fundamentales para implementar la navegación en Flex es la clase ViewStack.

Un contenedor **ViewStack** es una clase que se compone de una colección de contenedores secundarios que se apilan en la parte superior de cada uno, con un solo contenedor visible o activo a la vez. El contenedor ViewStack no define un mecanismo integrado para los usuarios para cambiar el envase activo actualmente, debemos utilizar para esto un LinkBar, TabBar, ButtonBar o ToggleButtonBar control o construir nuestra lógica propia de navegación en ActionScript para permitir a los usuarios cambiar la vista activa del ViewStack.

ViewStack solo puede tener a otros contenedores como nodos descendientes incluyendo otros componentes ViewStack.

Siendo ViewStack el elemento indispensable para la navegación en Flex, se requiere de otros componentes para cambiar la visibilidad de un nodo y otro en el componente ViewStack como puede ser un LinkBar, Button, etc.
Para trabajar con ViewStack podemos utilizar dos características muy útiles como son:

SelectedChild: Si queremos indicar cuál de los nodos descendientes de ViewStack tendría que visualizarse con un nombre lógico en lugar de un índice numérico. Esta propiedad visualizará el contenedor apropiado en ViewStacker basándose en el nombre de la instancia proporcionado por la propiedad id.

SelectedIndex: Propiedad que usaremos para elegir que nodo descendiente de ViewStack tendría que visualizarse.

La clase ViewStack no tiene un mecanismo incorporado para controlar qué contenedor descendiente se visualiza o se activa. Flex proporciona varias herramientas para controlar este aspecto, una de ellas es TabNavigator.

Lo que haremos a continuación será implementar un TabNavigator en nuestra aplicación Mantenimiento de Productos para seleccionar entre ingresar, modificar o eliminar un ítem.

1. Lo primero que haremos es crear una instancia de los dos componentes personalizados: uno para actualizar y eliminar productos y otro para añadir productos. ActualizaElimina.mxml es una subclase de la clase HBox y Añadir.mxml es una subclase de la clase VBox, esto es importante porque para utilizar TabNavigator con estos componentes personalizados, éstos tienen que ser contenedores.

2. Al final de los dos componentes personalizados pondremos:

```
<v:ActualizaEliminapro unidades="{unidades}"
        foodColl="{foodColl}"
    productActuali="showPopUp(event.product,'Actualiza
    Producto')"
    productElim="showPopUp(event.product,'Elimina Producto')"/>

<v:AnadeProduct
        cats="{categoria}" unidad = "{unidad } "
    productAnade="showPopUp(event.producto,'Producto
  Añadido')"/>
```

3. Rodeamos las instancias de ambos componentes con un conjunto de etiquetas:

<mx: TabNavigatorX Configure Wídth (Ancho) a 700 y Height (Altura) a 600.

Ejecutamos la aplicación y veremos:

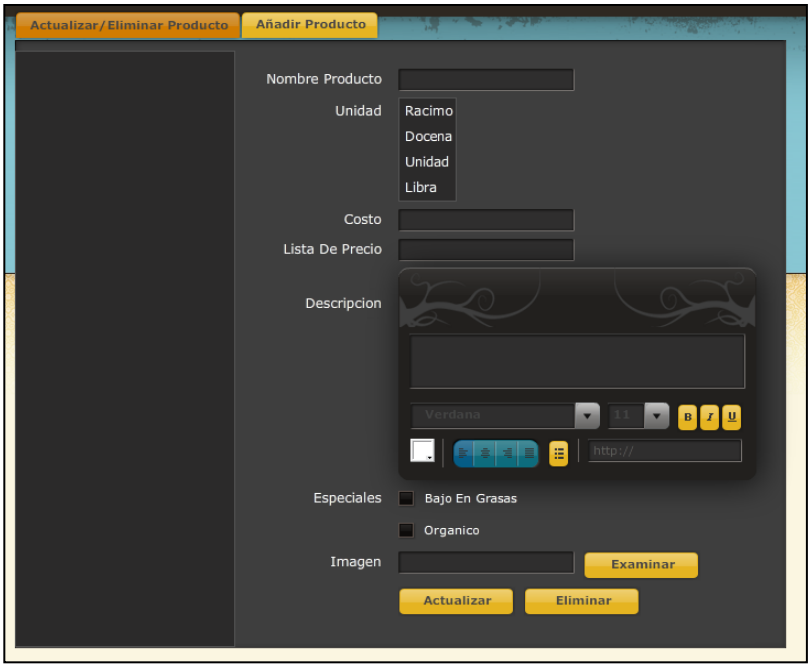


Figura 6. 33 Pestañas de la Aplicación Mantenimiento de Productos

Ahora se procederá a construir una página principal de QuickDelivery, un proceso de verificación utilizando ViewStack y Button para lograr navegar entre la página principal, la selección de productos y la pantalla de verificación.

Ahora llevaremos a cabo el proceso de validación de los datos de la compra usando los ViewStack, una vez que los usuarios tengan todos los productos que quieren en el carro de la compra, tendrán que verificarlos. El proceso básico es el siguiente:

- El usuario hace clic en un botón para cambiar a la página de verificación.
- El usuario rellena un formulario que proporciona la información básica de la factura y el envío como el nombre, la dirección, etc.
- El usuario hace clic en un botón en la página de información básica de factura y después pasa al formulario de la información de la tarjeta de crédito.
- El usuario rellena ahora el formulario proporcionando la información de la tarjeta de crédito.
- El usuario hace clic en el botón para comprar los artículos de la tienda de comestibles y después pasa a una página de confirmación de pedido.

Crearemos primero un objeto value haciendo clic con el botón derecho del ratón sobre la carpeta valueObjects y seleccione **New>ActionScript Class**. El nombre de la clase tiene que ser **InfoPedido.**

Esta clase contendrá toda la información sobre el pedido, incluyendo la información de facturación del usuario.

En la clase creamos las siguientes propiedades públicas utilizando los tipos de datos que aparecen a continuación.

- PedidoNNombre:String
- PedidoDirec:String
- PedidoCiudad:String
- PedidoProvincia:String
- TipoTarjeta:String
- NumeroTarjeta:Number
- cardExpirationMonth:Number
- cardExpirationYear:Number
- deliveryDate:Date

Una vez creada esta clase procedemos a crear el formulario de ingreso de datos para la compra y el resultado final será el siguiente:

Figura 6. 34 Verificación: Información cliente

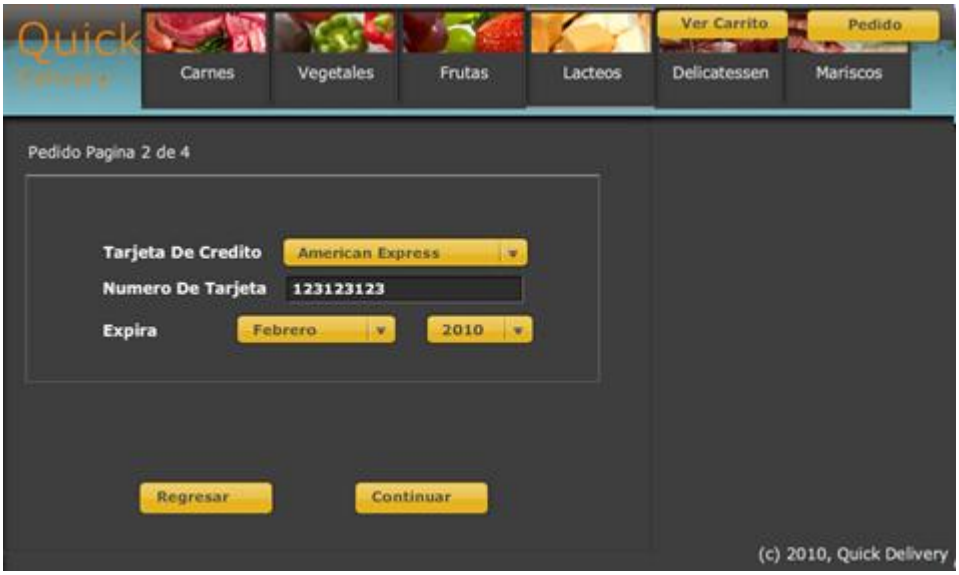


Figura 6. 35 Verificación: Información Tarjeta de Crédito

6.3.14.1 Manejo del historial del navegador

History management permite al usuario navegar a través de una aplicación utilizando los botones Back (atrás) y Forward (adelante) que como sabemos en las RIA no siempre son comunes. Por ejemplo un usuario puede hacer clic en una pestaña en TabNavigator o el contenedor Accordion y después moverse a la pestaña anterior haciendo clic en el botón Back (atrás) del navegador.

Flex admite de forma automática un history management para cualquier contenedor navigator sin utilizar ninguna etiqueta MXML o ActionScript y admite de forma predeterminada los contenedores Accordion y TabNavigator. El *history management* se inhabilita por defecto para el contendor ViewStack.

Podemos activar el administrador del historial para el contenedor *ViewStack* especificando a *historyManagementEnabled* y estableciendo ese atributo como true de la siguiente forma:

<mx: ViewStack. historyManagementEnabled="true">

Cuando se activa un *history management* el usuario se mueve de estado en estado dentro de la aplicación y se guarda cada estado. Cuando se selecciona el botón adelante o atrás en el navegador, el *history management* cargará y presentará el siguiente estado de la navegación que se ha guardado previamente.

Sólo se guarda el estado del contenedor navigator actual, si navigator tiene muchos nodos descendientes, los estados de estos componentes no se guardan a no ser que se active el *history management* para ese contenedor. En realidad, el *history management* guarda cada estado de la aplicación al utilizar la función de Adobe Flash Player navigateToURL( ) .

Esta función carga una estructura HTML invisible en la ventana del servidor actual. Todos los estados de la aplicación Flex se codifican en parámetros URL para la estructura HTML invisible. Un archivo SWF llama ahistory.swf, que está situado en la estructura invisible, decodifica los parámetros de la petición y envía la información del estado de navigation de vuelta a la clase HistoryManager en Flex, donde se visualizan los datos que se han guardado.

Podemos añadir el *history management* a cualquier componente personalizado en el que sea apropiado.

6.3.14.2 Deep Linking (Enlace profundo)

Deep linking (enlace profundo) es un concepto que describe que las aplicaciones que tradicionalmente son despachadas como un único UR.
A diferencia de las aplicaciones Web tradicionales basadas en HTML, las aplicaciones RIA creadas con Flex no son un conjunto desordenado de páginas que no están relacionadas y que tienen cada una su propio URL.

En Flex se pueda acceder a toda la aplicación a través de un único URL. Normalmente, cuando el URL cambia mientras una aplicación Flex se está ejecutando, el navegador descargará la aplicación y *cargará* la página que se ha solicitado. Esto hace que sea muy difícil enviar a un cliente un URL a una parte específica de la aplicación.
Deep linking permite al URL representar distintas "páginas" en una aplicación Flex sin tener que alojar la aplicación a medida que cambia el URL. Esto se consigue utilizando enlaces (los parámetros que siguen al "#" en un URL que se utilizan normalmente para poder navegar a distintos sitios dentro de una página HTML).

Cuando se implementa *deep linking* con enlaces, la parte principal del URL no cambia y, precisamente por ese motivo, el navegador no intentará descargar la aplicación y cargar una distinta.
En su lugar, lo único que cambia es el enlace, que puede ser leído y provocar una reacción de forma programática.
La parte del URL que se encuentra situada después de "#" se denomina fragmento.

Cada vez que cambia el fragmento, el fragmento anterior se almacena en el historial del navegador, permitiendo a los usuarios utilizar los botones adelante y atrás para avanzar y retroceder por la aplicación.

Ejemplo:

http://localhost/phpTesis/TQuickDelivery/QuickDelivery.html#categoria=0

Figura 6. 36 Ejemplo Deep Linking: Selección categoría Carnes

http://localhost/phpTesis/TQuickDelivery/QuickDelivery.html#categoria=1

Figura 6. 37 Ejemplo Deep Linking: Selección categoría Vegetales

6.3.15 Uso de Estilos para el Proyecto QuickDelivery

Durante el desarrollo de este capítulo hemos estado mostrando imágenes de cómo se está desarrollando el proyecto QuickDelivery mientras explicamos muchos de los componentes y funcionalidades de Adobe Flex 3, pero, por razones de presentación se han mostrados imágenes de una aplicación que posee formatos de estilos y pieles como lo vamos a mostrar, sin embargo en este subtema mostraremos como es la aplicación sin los estilos que presentamos y demostraremos como se aplica y configura el diseño de esta aplicación mediante Styles (estilos) y Skins (pieles).

Existen dos formas distintas para aplicar un diseño a nuestras aplicaciones Flex: *styles* (estilos) y *skins* (pieles). Podemos modificar la apariencia de cualquier componente Flex mediante el uso de las propiedades style, que se pueden utilizar para configurar el tamaño de la fuente, el color del fondo y muchas otras propiedades predeterminadas de estilo.

Otra opción para personalizar la apariencia de una aplicación Flex es utilizar *skins* (pieles), que son elementos gráficos (en forma de archivos o diseñadas con ActionScript) que pueden utilizarse para sustituir la apariencia predeterminada de los distintos estados del componente.

6.3.15.1 Aplicar estilos

La aplicación de comercio electrónico QuickDelivery que estamos construyendo cambia su apariencia por completo con styles (estilos) y skins (pieles). Como hemos visto hasta ahora, el desarrollo de Flex se efectúa con una serie de lenguajes basados en estándares como MXML (basado en XML) y ActionScript 3.0 (ECMAScript). Hay muchas formas distintas de aplicar un estilo como puede ser configurar un único estilo en un componente particular; utilizar selectores de clase CSS para aplicar varios estilos a la vez que puedan aplicarse a varios componentes; o utilizar un selector para especificar que todos los componentes de un tipo en particular deberían utilizarse como un conjunto de estilos.

Cualquier componente que tenga texto tiene los siguientes estilos:

- **Color:** Color de texto del componente, especificado como un número hexadecimal.

  El valor predeterminado es 0x0B333C.

- **disabledColor:** Color del componente si es desactivado, especificado como un número hexadecimal. El valor predeterminado es 0xAAB3B3.

- **fontFamily:** Nombre de la fuente utilizada y que se especifica como una cadena. Puede utilizarse cualquier nombre de fuente.

- **fontSize:** Altura del texto especificado en número de píxeles. El valor predeterminado es 10.

- **fontStyle:** Cadena que indica si el texto aparece en cursiva. Los valores reconocidos son normal e italic (cursiva). El valor predeterminado es normal.
- **fontWeight:** Cadena que indica si el texto aparece en negrita. Los valores reconocidos son normal y bold (negrita). El valor predeterminado es normal.

- **marginLeft:** Número de píxeles entre el borde izquierdo del contenedor y el borde izquierdo de su área de contenido. El valor predeterminado para los controles Text es 0, pero se aplican valores distintos dependiendo de los distintos componentes.

- **marginRight:** Número de píxeles entre el borde derecho del contenedor y el borde derecho de su área de contenido. El valor predeterminado para los controles Text es 0, pero se aplican valores distintos dependiendo de los distintos componentes.

- **textAlign:** Cadena que indica la alineación del texto dentro de su contenedor o control. Se reconocen los

valores left (izquierda), right (derecha) o center (centro). El valor predeterminado es left.

- **textDecoration:** Cadena que indica si el texto aparece subrayado. Se reconocen los valores none (ninguno) y underline (subrayado). El valor predeterminado es none.

- **textIndent:** Desplazamiento de la primera línea de texto desde el lado izquierdo del contenedor, especificado un número de píxeles. El valor predeterminado es 0.

En la siguiente imagen mostraremos la aplicación sin la utilización de estilos ni pieles:

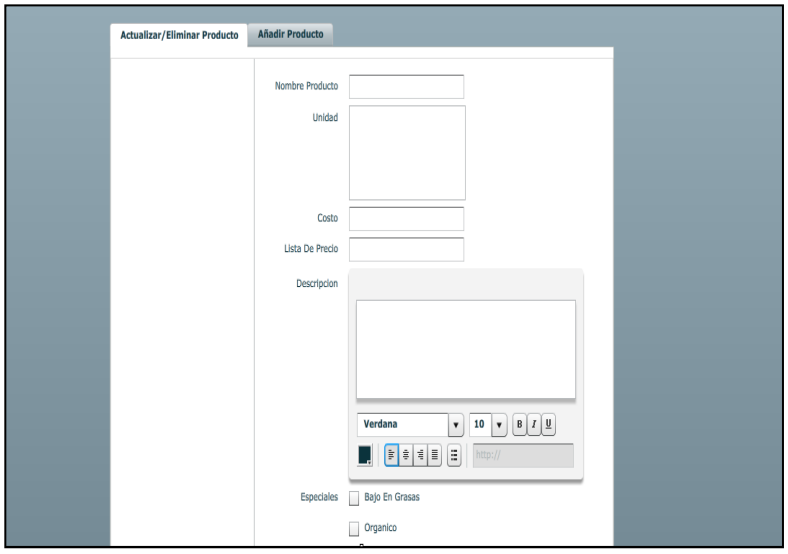


Figura 6. 38 Aplicación Mantenimiento de productos sin aplicar estilos

Para aplicar un determinado estilo en un proyecto de Flex es necesario añadir la etiqueta **<mx:Style/>** con la ruta del archivo donde se encuentra nuestra hoja de estilos css:

```
<mx:Style source="assets/css/Main.css"/>
```

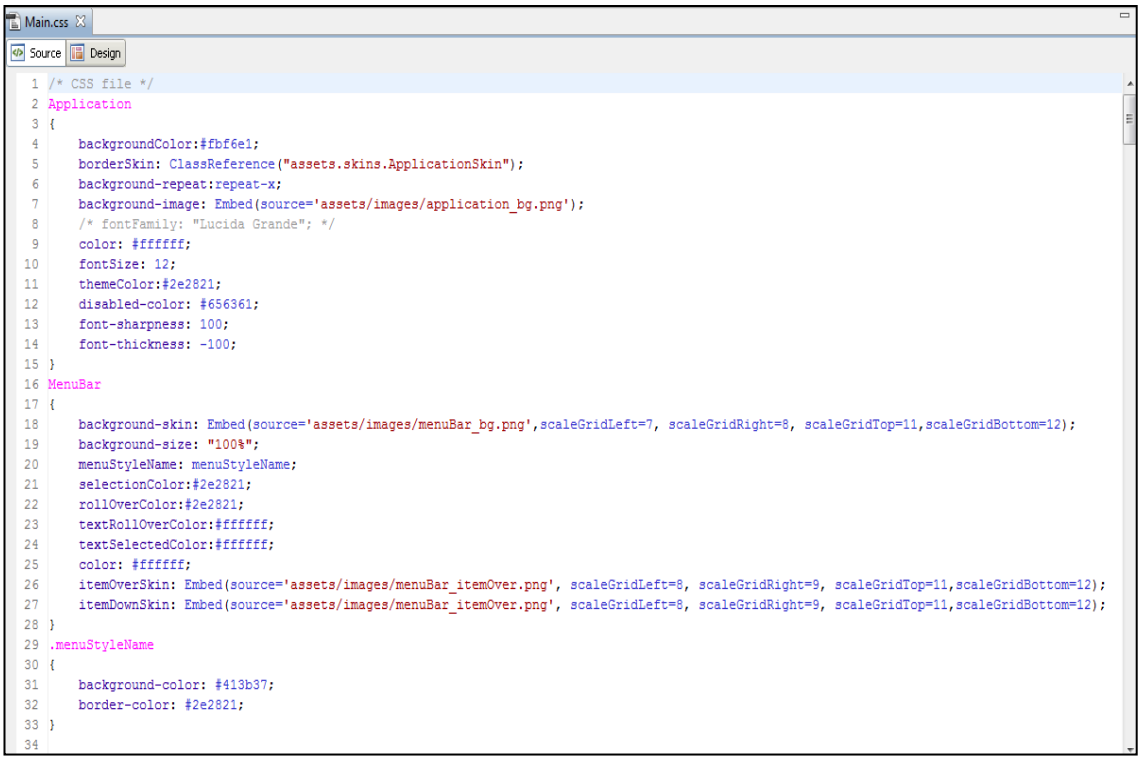


Figura 6. 39 Archivo Main.css para estilos de la aplicación

Otro potencial de esta herramienta radica en que Adobe ha añadido un sencillo mecanismo para convertir una hoja de estilo CSS ya existente en un SWF con el que Flash Player puede interactuar fácilmente.

Utilizando el SDK, puede utilizar el compilador MXMLC para compilar un archivo CSS en una vista Flex Navigator (Navegador de Flex) y seleccionar la opción Compile To SWF (Compilar a SWF), como aparece en la figura 6.40.

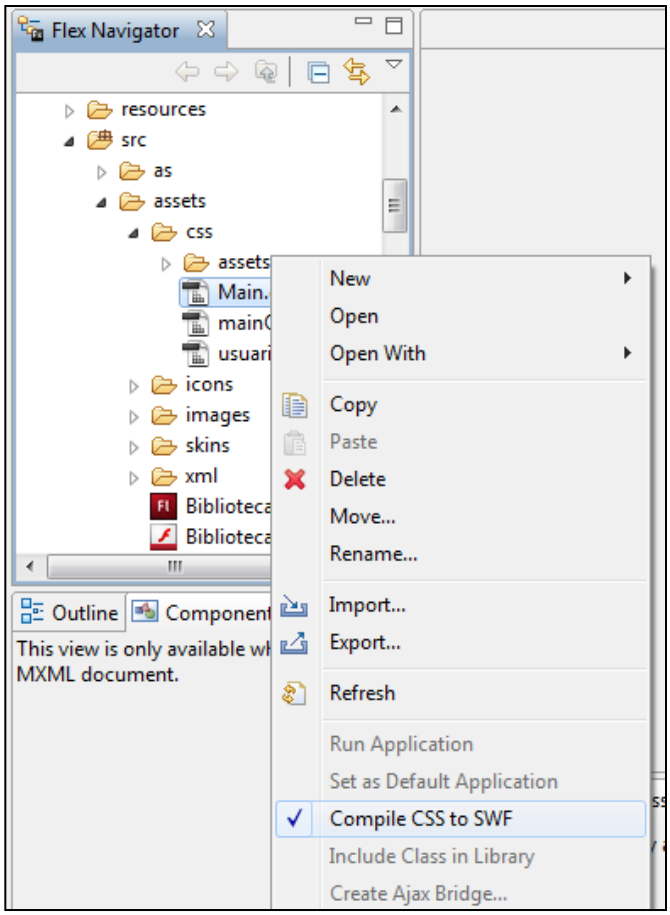


Figura 6. 40 Programar Flex 3 para que compile una Hoja de estilo en un archivo de Flash Player

6.3.15.2 Pieles Gráficas

A veces es necesario aplicar un diseño a una aplicación Flex. En estos casos vamos a utilizar *skinning* (pieles).

Las pieles se utilizan para cambiar la apariencia de un componente modificando o reemplazando sus elementos visuales. A diferencia de los estilos, que cambian los valores de las pieles existentes para un componente, las pieles nos permiten sustituir los elementos usados. Esto puede realizarse de forma gráfica con imágenes y archivos SWF o de forma programática con archivos de clases y la API.

Cada componente está formado por varias pieles que representan distintas apariencias visuales del componente. Por ejemplo, un botón tiene una piel distinta para su apariencia normal (upSkin), para su apariencia cuando el cursor se mueve sobre él (overSkin), para cuando se presiona (downSkin) y para cuando está desactivado (disabledSkin).

Una forma para aplicar pieles a un componente es especificar nuevos elementos gráficos que serán usados en vez de las pieles predeterminadas.

Por ejemplo:

1. En línea:

```
<mx:Button upSkin="@Embed('../assets/miSkinUpSkin.gif')"/>
```

2. Configuración en un bloque (o archivo) CSS:

```
<mx:Style>
Button {
overSkin: Embed("../assets/images/miSkinOverSkin.gif");
</mx:Style>
```

3. Configuración en ActionScript:

```
<mx:Script>
            [Embed("assets/miSkinDownSkin.gif")]
        var ds:Class;
    function initApp(){
                miBoton.setStyle("downSkin",ds); }
</mx:Script>
```

6.3.15.3 Pieles programáticas

En vez de usar recursos gráficos para las pieles, también es posible utilizar la API de dibujo de Flash Player (se encuentra en su mayor parte en la clase flash.display.Graphics para dibujar

de forma programática nuestra propia piel. La principal razón para elegir una piel programática en lugar de una gráfica es que podemos tener mucho más control sobre ella cuando se hace de forma programática. En lugar de integrar gráficos de un tamaño establecido, las pieles programáticas pueden construirse fácilmente para reconfigurar su propio tamaño, mientras que una piel gráfica no reconfigura su tamaño. Las pieles programáticas también usan menos memoria ya que no contienen gráficos externos.

La propiedad fillColors aplica igual cantidad de cada color, mientras que la API de dibujo nos proporciona un método beginGradientFill() que no sólo nos permite especificar el array de colores, sino también los porcentajes de cada uno, además de una matriz para determinar la dirección del degradado.

Flex proporciona tres clases entre las que podemos elegir:

- **ProgrammaticSkin:** Ejecuta las interface IFlexDisplayObject, ILayoutClient e IStyleable. Ésta es la clase con menos peso que podemos utilizar como superclase para una piel.
- **Border:** La clase Border amplía la clase ProgramaticSkin y añade soporte para la propiedad borderMetrics. Si estamos buscando poner en práctica una piel con un borde que no utilice una imagen de fondo, este es la mejor opción.
- **RectBorder:** La clase Rect Border amplía la clase Border y añade soporte para el estilo backgroundImage. Éste es el método utilizado para dibujar los elementos visuales de cualquier clase, de forma que crea una nueva apariencia para una piel.

6.4 Utilización de datos en la aplicación QuickDelivery

En esta parte hablaremos de cómo acceder a los servicios Web en un servidor para recuperar datos para nuestras aplicaciones. Para esto aprenderemos a usar el nuevo Web Service Introspección Wizard de Flex 3, así como la programación necesaria para poder trabajar con Servicios Web a través de ColdFusion, Php y MySQL.

6.4.1. Importancia de los Servicios de Consumo de Información

La necesidad de que los clientes realicen llamadas al servidor para el tratamiento de datos empresariales apareció la primera vez que múltiples usuarios quisieron acceder a la vez al sistema. Las llamadas se efectuaban sobre protocolos definidos y personalizados realizadas con llamadas a procedimientos remotos (RPC). Las primeras RPC se diseñaron en un principio para trabajar con una plataforma y un producto de un distribuidor específico. Tenían un formato específico para los datos que se transmitían. Hoy en día, el término RPC describe la llamada más básica entre un cliente y un servidor en lugar de a un protocolo específico o un formato de mensaje.

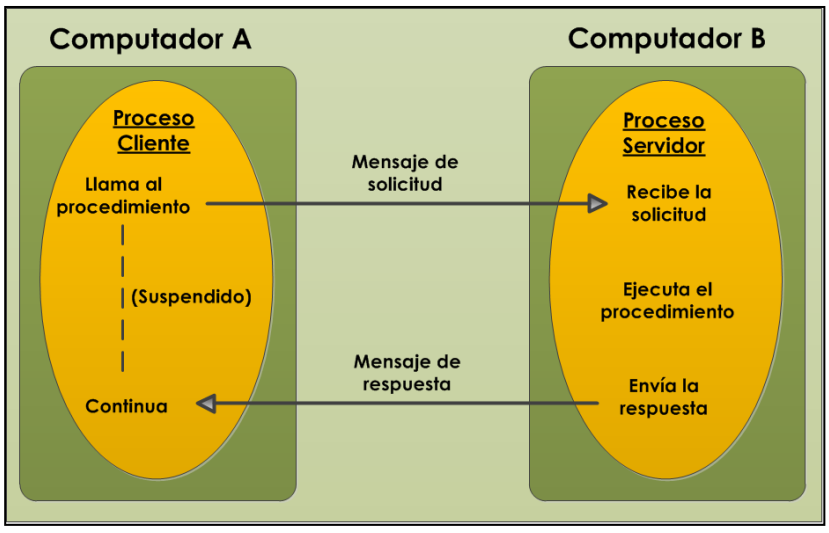


Figura 6. 41 Ejemplo del consumo de un servicio Web

La aparición de CORBA *(Common Object Request Broker Architecture,* Arquitectura común de intermediarios en peticiones a objetos) como un estándar de la industria para exponer los objetos del servidor en un nivel intermedio fue muy importante a la hora de compartir lógica entre aplicaciones.

La gran desventaja era la diferencia entre la ejecución de CORBA en los distribuidores y la planificación de los datos entre los diferentes sistemas. Java introdujo RMI *(Remote Method Invocation,* Invocación a un Método Remoto) como una forma más refinada para llamar a objetos remotos y transmitir los datos entre los niveles del cliente y del servidor. Pero, debido a la complejidad de ejecución, RMI no se ha usado por los desarrolladores fuera del mundo de Java. La forma más reciente de comunicación entre cliente y servidor son los ***Web Services*** (Servicios Web), que usan el XML basado en texto para describir cómo se utilizan los objetos del servidor además del formato en el que se realiza la comunicación.

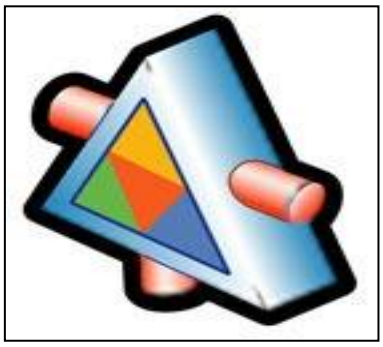


Figura 6. 42 Logo de un Web Service

Un Web Service se puede ejemplificar como una página Web dinámica leída por una aplicación cliente.

Aunque los servicios Web se utilizan (o se consumen) normalmente por una aplicación cliente, los datos y la descripción de esos datos permanece *human readable* (puede ser leída por humanos) como

XML. Sin embargo, esta ventaja también es una debilidad. El texto *human readable* no es tan eficiente como la comunicación binaria entre cliente y servidor.

En Flash MX, se introduce Flash Remoting como una forma para proporcionar comunicación binaria entre el cliente Flash y el servidor. Flash Remoting se basa en un protocolo abierto llamado AMF *(Action Message Format,* Formato de Mensaje de Acción) que nos permite comunicarnos con una variedad de tecnologías de servidores incluyendo Java, PHP y ColdFusion.

6.4.2 Flex Builder 3 y los Web Services

Al hacer llamadas al servidor, estamos llamando a la lógica que esta fuera de Flex. Al igual que en la etiqueta <mx: HTTPService>, no tenemos control cuándo el servidor terminará de ejecutar la petición que hemos realizado, por lo que necesitamos trabajar con estos eventos y escuchadores de eventos para capturar el momento en el que el servidor ha terminado con la petición.
Flex transmite un evento *result* si el servidor ha sido capaz de procesar la petición con éxito, lo cual significa que devuelve datos desde la consulta o sencillamente realiza una acción en un servidor. Por el contrario, se transmite un evento *fault* en caso de que se produzca un error. Tanto el evento *result* como el *evento fault* llevan información adicional, como cualquier dato devuelto o una indicación de por qué se ha producido un fallo.

6.4.3 Trabajando con ColdFusion

Como parte de este capítulo hablaremos de cómo trabajar con ColdFusion 8 y Flex Builder 3, y empezaremos haciendo una pequeña reseña en la que trataremos sobre la instalación y puesta en marcha de este servidor, cabe recalcar que no trataremos la parte teórica de este tema ya que esto lo hemos hecho en el capítulo 2.

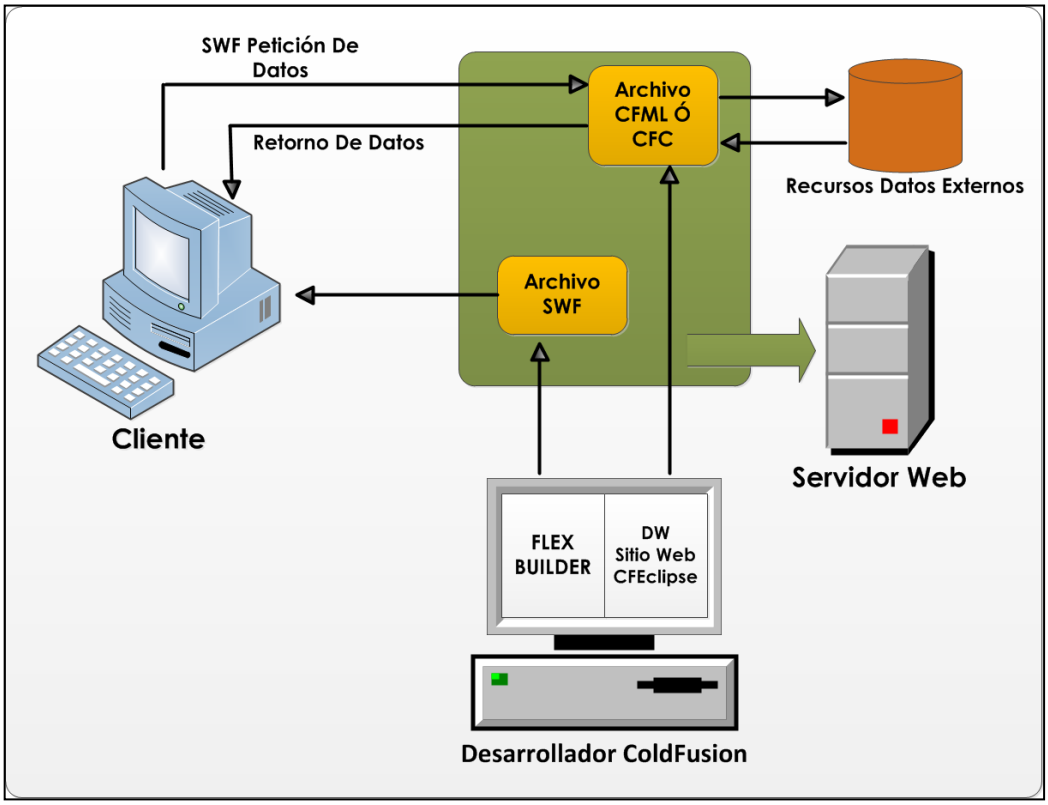


Figura 6. 43 Arquitectura Servidor ColdFusion

6.4.3.1. Instalación de servidor ColdFusion

Para nuestro proyecto usaremos JRun 4 de Macromedia, este es un servidor de aplicaciones Java cuyo objetivo es ofrecer una rápida y confiable plataforma compatible con J2EE. JRun se diseñó en el año 2002 y comprende una solución accesible y completa para el desarrollo y despliegue de aplicaciones Java

robustas con un buen desempeño en funcionamiento y tiempo de ejecución.

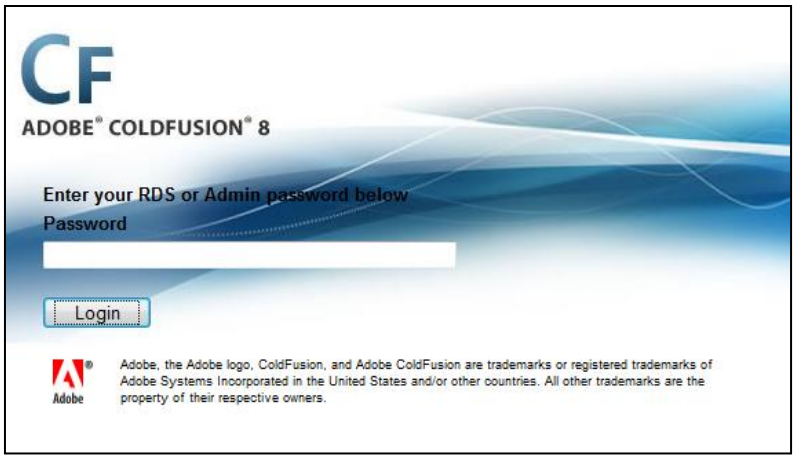


Figura 6. 44 Servidor ColdFusion 8

Es sobre esta versión de JRun 4 que correremos nuestro servidor ColdFusion el mismo que lo instalaremos de la siguiente manera:

1. Una vez descargado el directorio "cfusion", copiamos esta carpeta en la raíz de la unidad C: de nuestra PC.
2. Descomprimimos los archivos que se encuentran en esta carpeta y procedemos a ingresar en la ruta:

   C:\cfusionFlexTFS\bin

   Es aquí donde se encuentra los ejecutables de nuestro servidor de aplicaciones de donde podemos poner en funcionamiento el servidor ColdFusion.

3. Abrimos el terminal de comandos DOS y vamos a la ruta antes mencionada y mediante el siguiente comando ejecutamos la aplicación JRun 4:

   Jrun-start cfusion

4. Comprobamos que no existen errores durante el inicio de JRun 4 y procedemos a encender nuestro servido de ColdFusion.

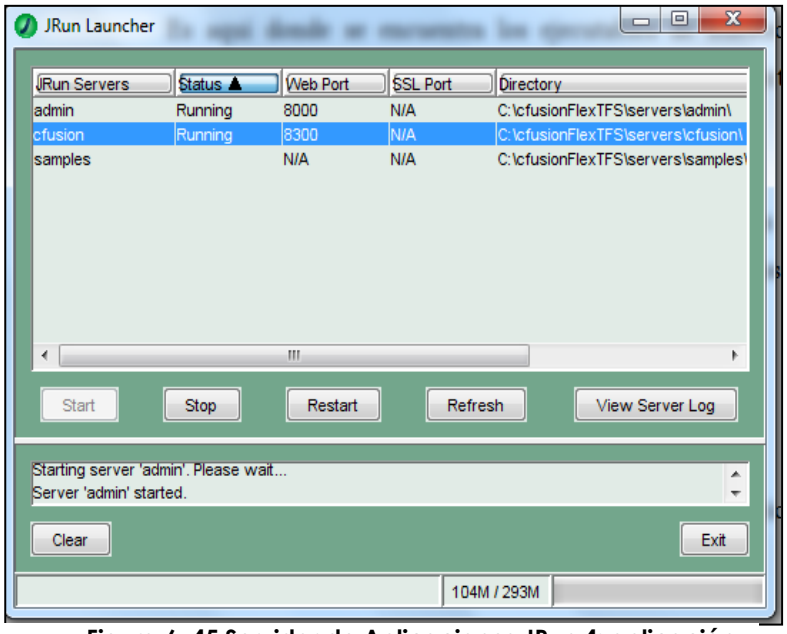


Figura 6. 45 Servidor de Aplicaciones JRun 4: aplicación sobre la que se ejecuta el Servidor ColdFusion

Desde ahora trabajaremos sobre el servidor local ColdFusion para todos los datos y los objetos remotos que utilizará la aplicación. Configuraremos el puerto de comunicación con este servidor el número 8300.

Lo primero que haremos será explicar mediante un sencillo ejemplo, cómo crear un servicio Web utilizando ColdFusion.
Vamos a crear una aplicación a la que llamaremos bienvenidos, la que nos mostrará un pequeño mensaje de bienvenida.

Creamos el archivo "bienvenidos.cfm" y en su interior escribiremos el siguiente código.

```
<CFSET Var = "Bienvenidos">
<CFOUTPUT>#Var#</CFOUTPUT>
```

La etiqueta <CFSET> asigna un valor a una variable. Los nombres de variables deberían empezar siempre con una letra y pueden tener letras, números o guion bajo. En ColdFusion no se requiere asignar un tipo a una variable:

```
<CFSET Var = 1100>
<CFSET Var = TRUE>
<CFSET Var = "Juan">
<CFSET Var = "Tesis de " & Var>
```

La etiqueta <CFOUTPUT> </CFOUTPUT> y las almohadillas # las usaremos para mostrar el valor de una variable.

```
<CFOUTPUT>#Var#</CFOUTPUT>
```

El resultado de este ejercicio sería:

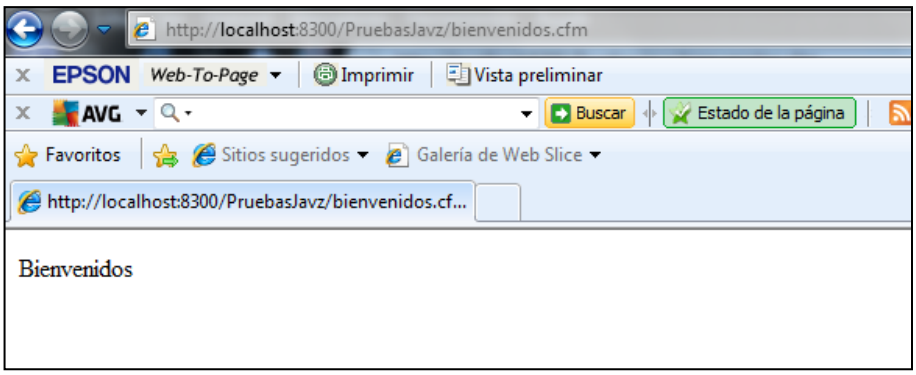


Figura 6. 46 Ejemplo simple de trabajo con ColdFusion

Sólo tenemos que consumir el servicio web y obtener los datos deseados.

6.4.5 Trabajando con Web Services Introspection Wizard

En esta sección utilizaremos el asistente para generar código y acceder a estos métodos del servicio Web. El código generado proporciona todos los beneficios de las operaciones codificadas con MXML y la flexibilidad de una instancia creada en ActionScript.

1. En el menú Data (Datos) de Flex Builder, seleccionamos Import Web Service (WSDL) (Importar servicio Web (WSDL)).

2. Seleccionamos la opción FlexGrocer [main source folder] (FlexGrocer [carpeta de origen principal]) y hacemos clic en Next.

3. Seleccionamos la opción Directly from the client (Directamente desde el cliente) como el método que utilizará la aplicación para acceder al servicio Web.

4. A la derecha de esta opción, el asistente de Flex Builder indica que esto requiere un archivo *crossdomain.* Un archivo *crossdomain* como lo mencionamos antes, es un pequeño archivo XML que existe en el mismo servidor en el que existe el archivo WSDL para este servicio Web. Su función es configurar la política para los clientes que utilizan Flash Player para acceder a los recursos en el servidor. Permite al autor o administrador del servidor Web aceptar o denegar el acceso basándose en el origen de la película Flash.

5. Especificamos el campo WSDL URL como http://localhost: 8300/tesisFlex/cfc/mantenimientoProductos.cfc?wsdl, que es la misma dirección que hemos usado para referirnos a nuestro servidor de ColdFusion. Hacemos clic en Next.

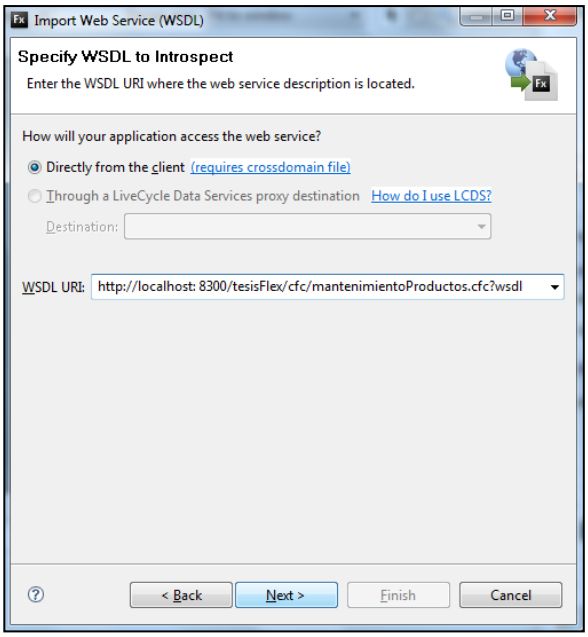


Figura 6. 47 Opciones del menú Import Web Service

6. El asistente carga el archivo WSDL desde el URL que se ha especificado y devuelve una lista de las operaciones disponibles, los argumentos que requieren cada operación y el tipo de retorno esperado.

7. Dejamos todas las operaciones activadas y Package name como generated.webservices y que Main class sea MantenimientoProductos.

8. Estos parámetros le dicen al asistente que tiene que crear una nueva clase llamada MantenimientoProductos en el paquete generated.webservices que tendrá métodos para actualizar, obtener, eliminar y añadir productos a través de este servicio Web.

9. Hacemos clic en Finish (Terminar).

10. El asistente genera código para facilitar la comunicación con los servicios Web en su nombre.

11. Una vez iniciado ColdFusion, procederemos a crear la conexión necesaria en Adobe Flex 3 utilizando la etiqueta <mx:HTTPService> como en el ejemplo:

```
<mx:HTTPService id="catRPC"
    url="http://localhost:8300/flexTesis/xml/categoria.xml"
    result="catHandler(event)"/>

<mx:HTTPService id="prodByCatRPC"
        url="http://localhost:8300/flexTesis/xml/categoricionProd
        uctos.cfm"
            result="prodByCategoryHandler(event)"
            resultFormat="e4x"/>
```

Hay dos formas distintas de llamar a un servicio Web en Flex. La primera está basada en etiquetas, la otra es a través de ActionScript. Utilizaremos la etiqueta <mx: WebService> para crear un objeto Web Service hacia el que podamos llamar nuestros métodos.

Creamos un bloque Web Service para la aplicación QD Estadísticas y ponemos una única etiqueta <mx: WebServicex>. Configuramos el atributo ID a EstadisticaWS. Configuramos luego el atributo wsdl a http://localhost:830O/flexGrocer/cfcs/aggregate.cfc?wsdl. Finalmente, dejamos el manejador fault de la misma forma que lo utilizaba <mx :HTTPService>.

El parámetro wsdl de la etiqueta Web Service especificamos el URL en el que Flex puede encontrar la información sobre las prestaciones de este objeto del servidor.

Se define en Web Service Description Language (WSDL).

Flex cargará este WSDL para entender los métodos disponibles en el servidor y cómo se puede acceder a ellos. Una vez que el archivo WSDL está cargado con éxito, la etiqueta WebService transmite un evento llamado load.
En Flex podemos utilizar los métodos de un servicio Web de estas formas:

- Fully Declared Method
- Declared Method

6.4.6 Flex y PHP con la herramienta Flex Builder Data Wizard

Flex Builder 3 incluye algunos asistentes de datos que nos ayudan a crear la parte del cliente y del servidor de una aplicación de forma sencilla. Estas aplicaciones sirven como una herramienta de aprendizaje muy útil para entender los puntos más importantes de la comunicación del cliente. Para esto realizaremos los siguientes pasos:

Nos dirigimos al menú de la aplicación principal seleccionando Data>Create Application from Datábase (Datos>Crear aplicación desde base de datos) en Flex Builder 3.
Este menú nos demuestra el potencial de Flex 3 para trabajar con servidores de la clase de PHP, ASP, .NET o J2EE (cuando se utilizan con LiveCycle Data Services o BlazeDS, un servidor Java remoto y de mensajería de código abierto distribuido por Adobe).

Estos asistentes sólo necesitan unos pocos pasos para generar una aplicación, teniendo en cuenta que ya tenemos uno de los entornos mencionados anteriormente para configurarlos y ejecutarlos correctamente en nuestra PC.

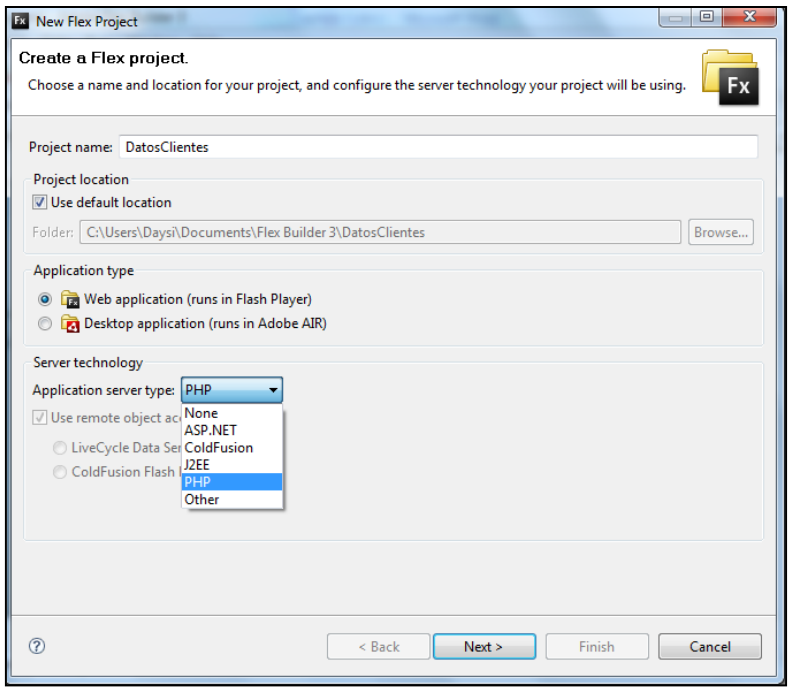


Figura 6. 48 Tecnologías de Servidores de Adobe Flex 3

Mostramos la pantalla inicial de F B Data Wizard:

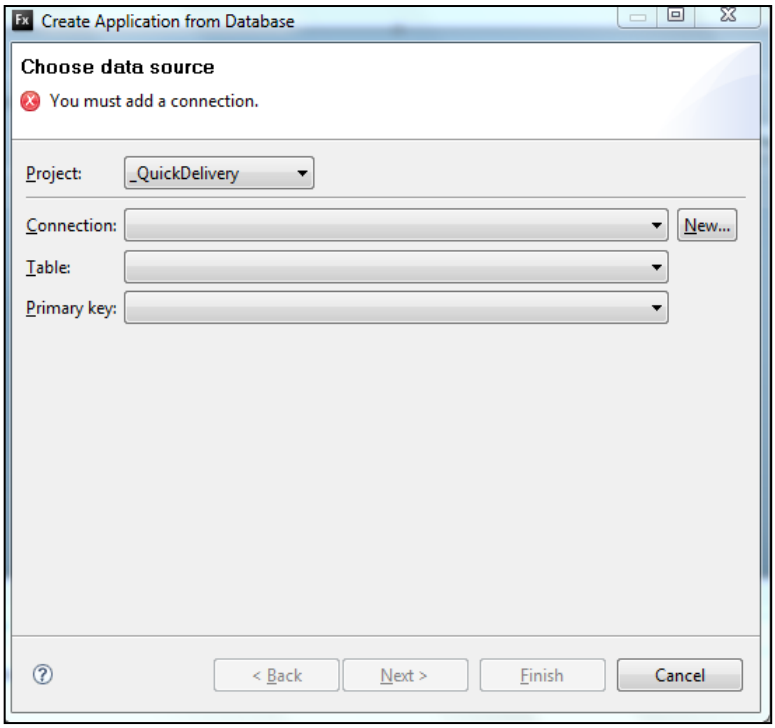


Figura 6. 49 Data Wizard: Configuración de conexión

Para este ejemplo crearemos una nueva conexión a nuestra base de datos y le daremos los detalles de conexión a nuestro servidor Apache / PHP y a nuestra base de datos de MySQL a la que por ejemplo llamaremos Tesis.

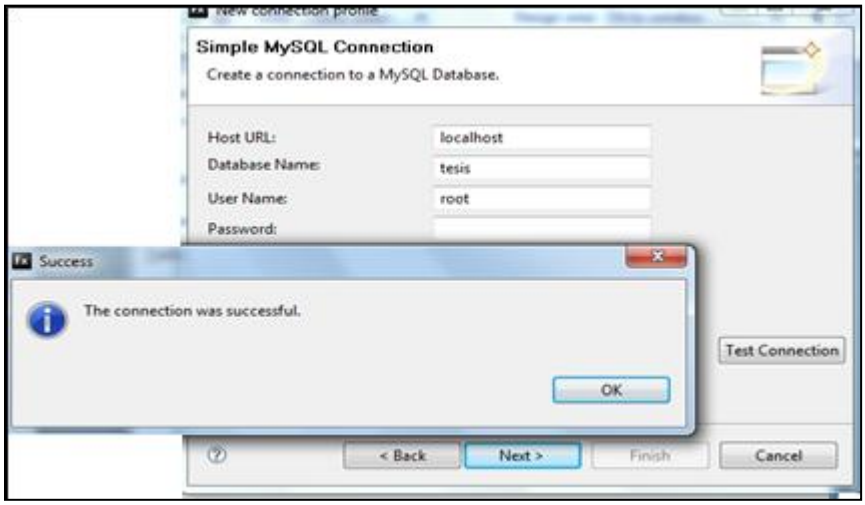


Figura 6. 50 Pantalla de creación de la conexión con MySQL

Testeamos la conexión, y verificamos su funcionalidad al recibir el estatus de Succces como lo demuestra la Figura 6.50. Lo siguiente por hacer es terminar la configuración de los parámetros de conexión y base de datos como se demuestra:

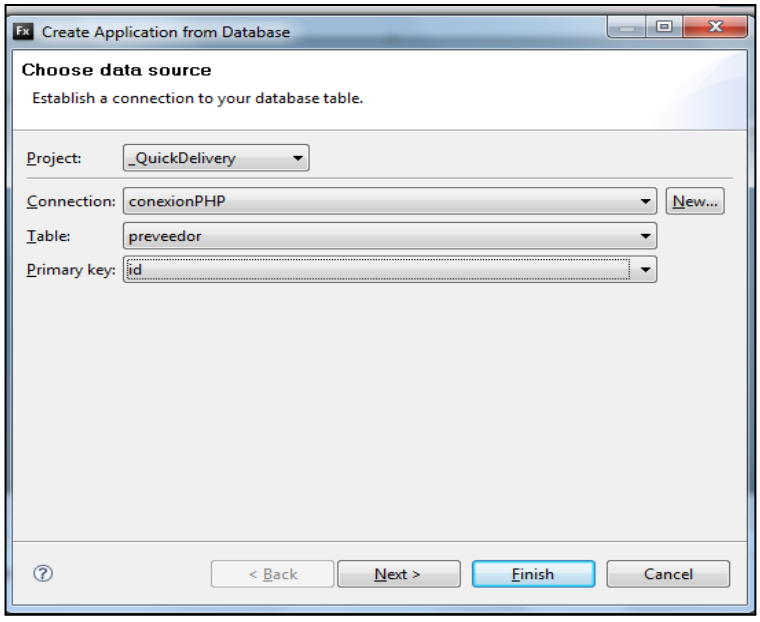
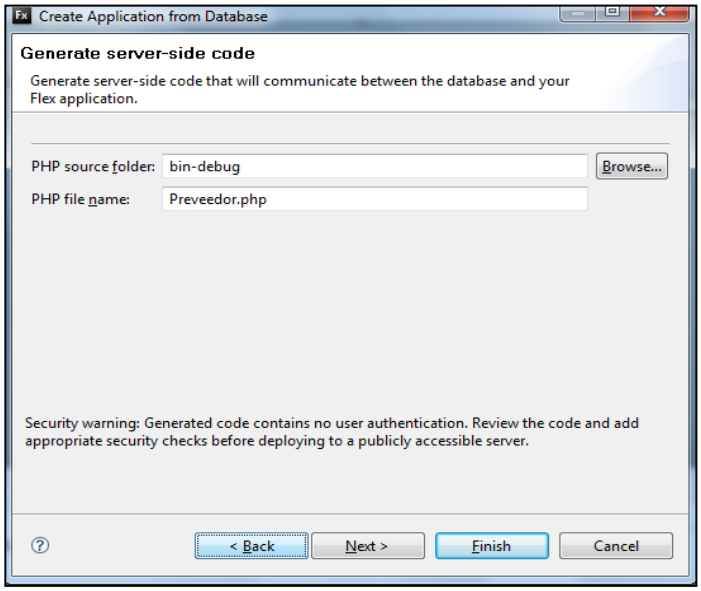


Figura 6. 51 Parámetros de conexión con la Base de Datos

Ahora configuramos los archivos de conexión entre nuestra aplicación de Flex 3 y PHP para que se generen los archivos que realizaran el mantenimiento de nuestra aplicación.



Fiaura 6. 52 Pantalla de confiauración archivos de conexión Flex PHP

Damos clic en siguiente y nos muestra una pantalla en la que se puede apreciar la parte de cliente de nuestra aplicación, damos clic en finalizar y se habrá creado nuestra aplicación de mantenimiento de proveedores:

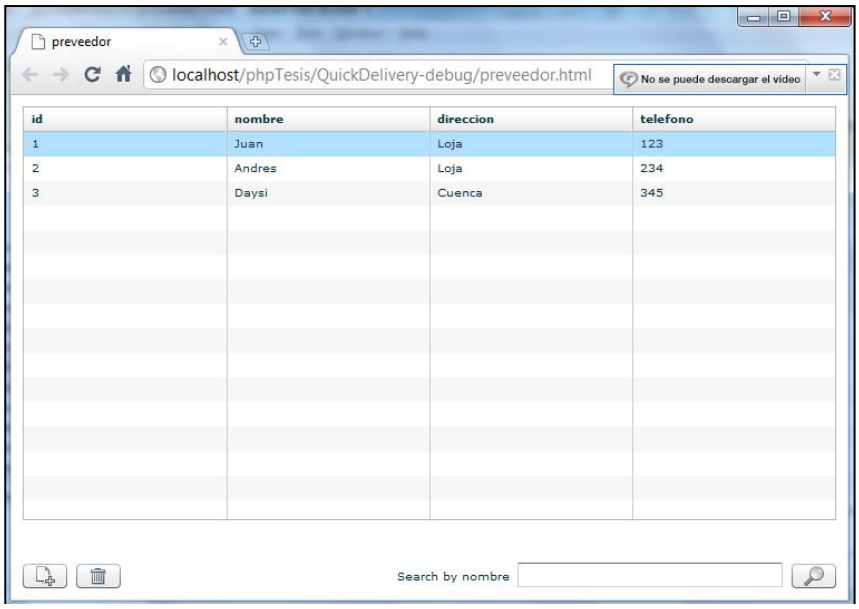


Figura 6. 53 Pantalla de la aplicación para Mantenimiento de Proveedores

Como pudimos observar en estos ejemplos, Flex es una herramienta muy potente a la hora de gestionar el trabajo con Datos de un servidor, sea este uno en ColdFusion como Apache con PHP, sea trabajando con Base de Datos o Web Services.

6.5 Creación de gráficos estadísticos para Quick Delivery

En este parte trataremos de hacer una presentación de los gráficos estadísticos de las ventas que se realicen de nuestro sitio Web. La visualización de los datos cuando se la realiza por medio de un gráfico es mucho más fácil de comprender para cualquier usuario. Por medio de Flex en lugar de presentar una tabla simple con datos numéricos podemos presentar un gráfico de barras, circular, de líneas u otro tipo de gráfico utilizando diferentes características.

Flex permite crear algunos de los tipos más comunes de gráficos bidimensionales, como gráficos de barras, de columnas y gráficos circulares, y nos permite un mayor control sobre la apariencia estos.

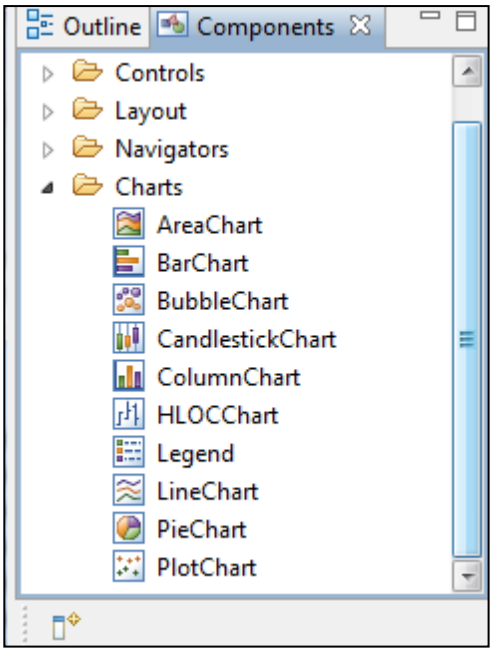


Figura 6. 54 Componentes Charts para trabajar con Gráficos Estadísticos

Los componentes Charting de Flex nos permiten crear los tipos de gráficos más comunes y también nos proporcionan un gran control sobre la apariencia de nuestros gráficos.

6.5.1 Tipos de gráficos a usar en Quick Delivery

La aplicación QD Estadísticos presenta gráficos ricos e interactivos de las ventas de una empresa. Para esto Flex Builder 3 proporciona varios tipos de gráficos, como los siguientes:

- Gráficos de Área.
- Gráficos de Barras.
- Gráficos Bubble.
- Gráficos Candlestick.
- Gráficos de Columnas.
- Line.
- Plot.
- PieChart.

Para nuestro proyecto usaremos los gráficos tipo Columnas, Line y PieChart.

6.5.2 Partes de un Gráfico

Existe un número de etiquetas distintas que podemos utilizar para definir un tipo de gráfico en concreto, pero en general la mayoría de los gráficos siguen esta estructura:

```
<ChartType>
        <!— Para Definir los ejes. —>
        <mx:horizontalAxis>
        <mx:AxisType/>
        </mx:horizontalAxis>
        <mx:verticalAxis>
        <mx:AxisType/>
        </mx:verticalAxis>
        <mx:series>
        <mx:SeriesName/>
        </mx:series>
<!— Para Dar estilo a los ejes y las marcas. -->
        <mx:horizontalAxisRenderer/>
        <mx:verticalAxisRenderer/>
<!—Añadir grid lines y otros elementos al gráfico. -
        <mx:annotationElements/>
        <mx:backgroundElements/>
</mx:ChartType>
<mx:Legend/>
```

Con este código podemos ver que un gráfico puede contener un tipo de gráfico, una o más series, uno o más ejes, *renderers* y otros elementos.

La serie define un array de clases Series que se utiliza para especificar los datos que van a ser interpretados en el gráfico. Para cada serie de datos se pueden utilizar estilos específicos, para personalizar cómo van

a ser interpretados los datos. Se proporciona un array de Series, por lo que un gráfico puede contener múltiples conjuntos de datos, tantos como necesitamos implementaremos en la comparación de gráficos, interpretando en gráficos tanto las ventas netas como brutas.

Cada tipo de gráfico tiene su propia clase Series que puede ser utilizada; por ejemplo, la serie para <mx: ColumnChart> es <mx: ColumnSeries>, mientras que la serie para <mx: PieChart> es <mx: PieSeries>.
La clase Axis se necesita para cualquiera de los gráficos rectangulares en dos dimensiones (también conocidos como gráficos cartesianos) porque especifica qué datos se interpretarán en el eje horizontal y vertical del gráfico. Existen subclases Axis que nos permiten especificar si Axis es dato numérico (LinearAxis) o está basado en una cadena (CategoryAxis).

6.5.3 Configurar los gráficos para QD Estadísticas

Ahora usaremos los componentes Charting para configurar las bases de los tres tipos de gráficos: gráficos de tipo, gráficos de ventas y gráficos comparativos.
Por el momento cada archivo de estructura sólo contiene un componente DataGrid con datos que se refieren a los productos vendidos. Ahora comenzaremos a añadir gráficos para visualizar estos datos. Trabajaremos en la construcción del programa QD Estadísticos.

Una vez creada la aplicación usamos el contenedor <mx: VBox> en <mx:ViewStack> añadimos <mx: PieChart> con Height (Altura) y Width (Ancho) configurados al 100% y dataProvider vinculado a dp.

```
<mx:ViewStack id="chartBase"
        Width="100%" height="100%">
```

```
<mx:VBox width= "100%" height="100%">
<mx:PieChart id="chart" width= "100%"
height="100%" dataProvider = "{dp}"
</mx:ViewStack>
```

Lo siguiente que tendremos que hacer es especificar las series antes de que nuestros datos puedan ser interpretados.

6.5.4 Rellenado de los gráficos de QD Estadísticas

Todos los gráficos necesitan que se especifique una o más series de datos. Los gráficos cartesianos también necesitan que especifiquemos el eje horizontal y vertical.

Ahora definiremos una serie, esto es un conjunto de datos que se proporciona a un gráfico. Dependiendo del tipo de gráfico que se utilice, se pueden usar distintas clases Series, a continuación se mencionan algunas:

- AreaSeries
- AreaSet
- BarSeries
- BarSet
- Bubbie series
- CandleStickSeries

Las que utilizaremos en la creación de nuestros gráficos estadísticos de ventas se muestran en la siguiente tabla:

Nombre De La Serie Clase	Descripción
ColumnSeries	La clase ColumnSeries define una serie de datos para un control **ColumnChart**
ColumnSet	ColumnSet es un conjunto de grupos que puede utilizarse para apilar o agrupar ColumnSeries en cualquier gráfico arbitrario
LineSeries	La clase LineSeries define una serie de datos para un control **LineChart**
PieSeries	La clase PieSeries define una serie de datos para un control **PieChart**.

Tabla 6 Tipos de gráficos para QD Estadística

Para especificar la serie de un gráfico añadiremos las siguientes funciones:

```
<v:VentasChart id="Ventas"
        width="100%" height="100%"
        title="Cuadro De Ventas"
        grossOrNet="{grossOrNetGroup.selection.data}"
        maximize="this.currentState='fullSales'"
        restore="this.currentState=''">
</v:VentasChart>
<mx:VBox id="rightCharts"
        width="100%" height="100%" >
```

```
<v:TipoChart id="type"
        width="100%" height="100%"
        title="Cuadro Por Categorias"
        grossOrNet="{grossOrNetGroup.selection.data}"
        typeChange="doTypeChange(event)"
        maximize="this.currentState='fullType'"
        restore="this.currentState=''">
</v:TipoChart>
<v:ComparacionChart id="comp"
        width="100%" height="100%"
        title="Cuadro De Comparacion"
        maximize="this.currentState='fullComp'"
        restore="this.currentState=''">

</v:ComparacionChart>
        </mx:VBox>
```

Con el código que hemos mostrado, estamos definiendo las clases necesarias para generar las series de datos que usaremos para poder mostrar el cuadro de ventas y comparaciones entre las diferentes categorías de productos vendidos.

Figura 6. 55 Gráfico circular (Pie Chart) de categorías de Productos

6.5.5 Ejes Horizontales y Verticales

Después de haber proporcionado los datos a los gráficos vamos redefinir los ejes utilizando para ello las etiquetas <mx:horizontalAxis> y <mx:verticalAxis>. Estas etiquetas pueden utilizarse para especificar los rangos válidos para el gráfico y también para planificar los datos en el gráfico. Flex admite cuatro tipos de ejes:

- **CategoryAxis:** Define una propiedad particular de los objetos del gráfico en el eje. Por ejemplo, un gráfico que presente la inscripción de una escuela basándose en la demografía de los estudiantes.

- **LinearAxis:** Define los datos numéricos en los puntos de un eje. Esto puede ayudarnos a especificar fácilmente los rangos numéricos válidos, los números que hay que omitir, etc.

- **LogAxis:** Define los datos logarítmicos de un eje. Para facilitarlo el eje Log tiene etiquetas para cada potencia de 10 (1,10,100,1000, etc.).
- **DateTimeAxis**: Define los valores basados en el tiempo en un eje. También puede utilizarse para configurar el formato de las etiquetas.
  Este eje también le permite desactivar fechas concretas con el fin de que no se visualicen, permite la creación fácil de un eje que visualice sólo los días de trabajo o los fines de semana, por ejemplo.

Ahora podremos tener varios ejes para un gráfico determinado, lo cual nos permite tener varios conjuntos de datos, unos junto a otros. Para facilitarlo, se crearon ejes adicionales dentro de una serie, como aparece en el siguiente código:

```
<mx:LineChart id="linechart" dataProvider="{myData}">
<mx:horizontalAxis>
        <mx:CategoryAxis id="catAxis" categoryField="Nombre de
        Categoria"/>
</mx:horizontalAxis>
<mx:verticalAxis>
        <mx:LinearAxis id= "vl"/>
</mx:verticalAxis>
<mx:verticalAxisRenderers>
        <mx:AxisRenderer axis = "{vl}"/>
        <mx:AxisRenderer axis = "{v2}"/>
        <mx:AxisRenderer axis = "{v3}"/>
</mx:verticalAxisRenderers>
<mx:series>
<mx:LineSeries yField="fieldName" form="curve"
displayName="Display Name"
itemRenderer="mx.charts.renderers.CircleItemRenderer">
</mx:LineSeries>
<mx:verticalAxis>
        <mx:LinearAxis id = "v2" />
</mx:verticalAxis>
</mx:LineSeries>
<mx:LineSeries yField="Nombre Categoria " form="curve"
displayName="Nombre Categoria" itemRenderer= "mx.
charts.renderers.CircleItemRenderer">
<mx:verticalAxis>
        <mx:LinearAxis id = "v3" />
</mx:verticalAxis>
</mx:LineSeries>
</mx:series>
</mx:LineChart>
```

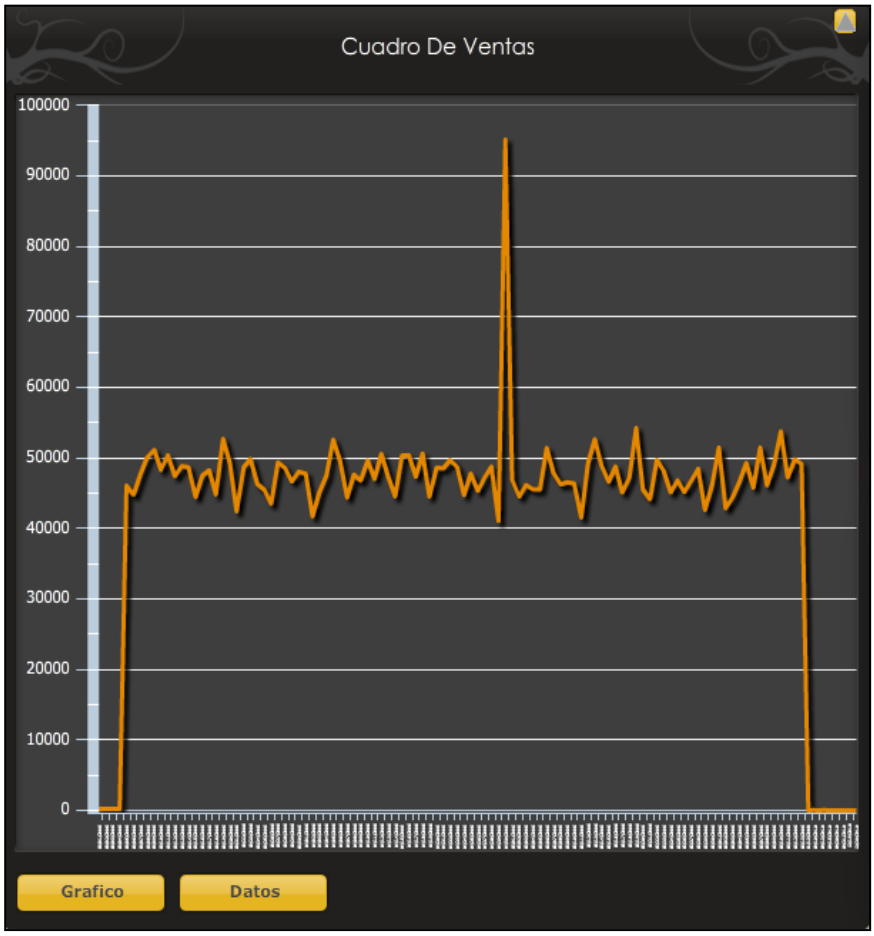


Figura 6. 56 Gráfico de líneas (Line Chart) de ventas de productos

6.5.6 Interacción con gráficos para QD Estadísticas

Entre los elementos que se ponen fácilmente en los gráficos están los *tips* (sugerencias de datos), cuando el usuario se mueva sobre los elementos del gráfico, además de permitir a los usuarios hacer clic en los elementos del gráfico para realizar otros cambios en la aplicación.
A continuación mostraremos algunos eventos que nos ayudaran a personalizar de mejor manera nuestros gráficos para QD Estadísticas.

Mouse Over: todos los gráficos admiten de forma inherente una propiedad llamada showDataTips. Al configurarla como true en un gráfico, aparecerá un elemento de tipo sugerencia y

presentará más información sobre el elemento sobre el cual el usuario se está moviendo.

Click: Otra interacción muy fácil de implementar es un evento c1ick, este evento lo usaremos para permitir al usuario hacer filtros en los gráficos de tipo y de comparación por categoría cuando el usuario haga clic en un sector del gráfico circular.

Selection: Además de la posibilidad de hacer clic en elementos individuales del gráfico, Flex 3 incluye la estructura para seleccionar varios elementos, bien manteniendo pulsada la tecla **Control** (o la tecla **Comando** en Mac) mientras se seleccionan los elementos o arrastrando un cuadro sobre varios puntos dentro del gráfico.

6.5.7 Añadir animación a los gráficos de QD Estadísticas

Entre la multitud de personalizaciones que vamos a llevar a cabo se incluye la posibilidad de tener datos animados en el gráfico o aplicar colores, degradados, etc. a los elementos del gráfico. Todas ellas están construidas como subclases de la clase mx.charts.effects.SeriesEffect. Estas clases pueden utilizarse con una serie showDataEffect o un atributo hideDataEff ect. Las clases son las siguientes:

- **SeriesInterpolate:** Este efecto mueve los gráficos que representan los datos existentes en una serie de puntos nuevos. En lugar de eliminar los datos del gráfico y luego volver a rellenarlo, crea una animación entre los puntos de datos originales y los nuevos.

- **SeriesSlide:** El efecto SeriesSlide desliza una serie de datos dentro o fuera de los bordes del gráfico. La propiedad direction especifica la situación desde la que se desliza la serie.

Gracias al uso de todas estas características podemos mostrar nuestros datos en una sola pantalla o alternar los resultados en una sola, pudiendo mostrarlos en gráficos estadísticos o datos en una tabla ejemplo:

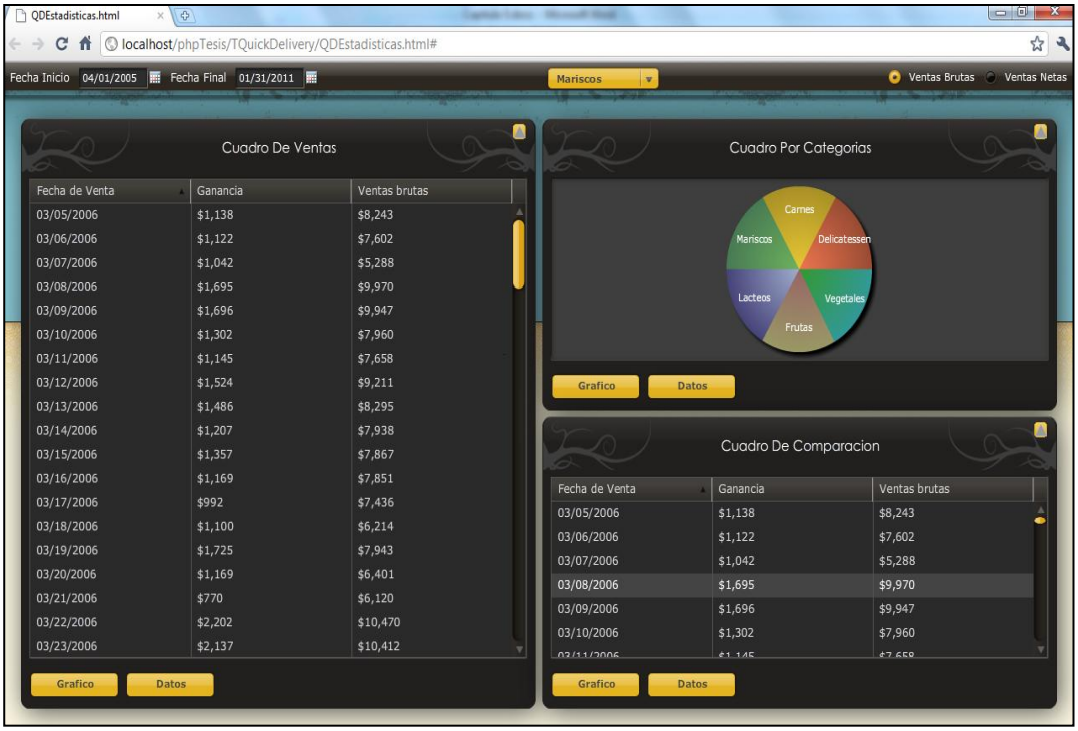


Figura 6. 57 Pantalla completa de la aplicación QD Estadísticas modo Datos

6.6 Modularidad en la creación de QuickDelivery

En esta sección trataremos de la importancia de crear aplicaciones modulares en nuestro proyecto, sabiendo que nuestro proyecto QuickDelivery está conformado por 3 aplicaciones que son: QuickDelivery (aplicación de comercio electrónico), QD Mantenimiento (mantenimiento de Productos), y QD Estadísticos (cuadros estadísticos y bodega de información), cada uno destinado a un usuario en particular.

Empezaremos dando nuestras razones para trabajar con aplicaciones modulares.

Los archivos al ingresar en él sistema se cargaban al iniciar la aplicación, creando imágenes SWF que se albergan en la memoria de nuestra máquina, y cuando se ejecutaba la aplicación no era necesario el acceso a Internet para recuperar estos archivos; esto era útil para crear aplicaciones sin conexión a Internet.

Sin embargo, esta práctica también tiene inconvenientes como es al crear imágenes en una aplicación aumenta el tamaño del SWF, haciendo que esta se cargue de forma más lenta. Es más, todos los usuarios experimentan la penalización de descarga de tamaño de estas imágenes independientemente de su utilización. Por ejemplo, si incrustamos todas las imágenes de QuickDelivery directamente en la aplicación, incluso un usuario que nunca vaya a la categoría Lácteos descarga la imagen de la botella de leche.

El código fuente para una aplicación Flex trabaja de una forma muy parecida. Por defecto, todos los archivos de clases utilizados por la aplicación, ya están escritos en MXML o en ActionScript, se crean en el archivo SWF. Esto tiene las mismas ventajas que una imagen incrustada: disponibilidad instantánea sin la necesidad de descarga. Sin embargo, también está sujeto a los mismos inconvenientes.

A medida que la aplicación crese aumenta su grado de complejidad, también aumentarán su tamaño y tiempo de descarga. Es más, un usuario que nunca utiliza una zona de la aplicación tiene que descargar las clases que proporcionan funcionalidad a esa zona.

Para nuestra ventaja, Flex proporciona una estrategia para crear módulos o lo que es igual dividir una aplicación extensa en partes más pequeñas. Con Flex Module Loader podemos cargar y descargar partes de la aplicación, dando a los usuarios un tamaño de descarga inicial más pequeño y permitiéndole encapsular y separar las partes de lógica de la aplicación.

Otro problema diferente, pero relacionado, al momento de desarrollar aplicaciones Flex es la duplicidad del código en distintas aplicaciones.

Flex presenta una solución para mejorar estos problemas conocida como "proyect Library" o Librería de Código.
Vamos a usar esta biblioteca para las aplicaciones actuales intentando solucionar estos inconvenientes.

Entrando un poco más en detalle en este problema, debido a que las aplicaciones son independientes, cada una crea su propia copia de algunas clases superpuestas como HBox, VBox, LinkButton. Esto castiga de forma innecesaria al usuario que utilice tanto la aplicación QuickDelivery como QD Estadísticas haciendo que tengan que descargar el mismo código muchas veces.

La solución a este problema se conoce como *Runtime Shared Librarles* (RSLs). Esto nos permite pasar a un plano exterior los recursos compartidos que podrían utilizarse en distintas aplicaciones y descargarlos sólo una vez, reduciendo el tamaño de descarga acumulado de varias aplicaciones.

6.6.1 Módulos en Flex

La sección de verificación de datos para la compra que está en la aplicación QuickDelivery es una buena opción para convertirla en un módulo de Flex Builder. Por un lado, una parte necesaria de la aplicación está dedicada a los usuarios que seleccionan comprar productos, los usuarios que sólo están mirando el contenido de nuestro sitio no necesitan este código adicional cuando se inicia la aplicación. Es más, una vez que esta sección tenga su propio módulo, se podrán realizar cambios a esta parte de la aplicación de forma separada del resto de la aplicación QuickDelivery.

Lo primero que haremos en Flex Builder es crear un nuevo directorio con el nombre **modules (**módulos**)** en el directorio raíz src de la

aplicación TQuickDelivery (nombre del proyecto de nuestra tesis) seleccionando **File>New>Folder** (Archivo>Nuevo>Carpeta) y especificando **modules** en el campo Folder name. Hacemos clic en **Finish** para crear la carpeta.

Esta carpeta dará la ubicación conveniente para almacenar cualquier módulo creado, separándolo del resto de la aplicación.
Luego crearemos la aplicación ModuloPedido.mxml desde el directorio src/views/QuickDelivery.

Veremos que la clase ModuloPedido actualmente tiene VBox, lo cual indica que la clase ModuloPedido utiliza un formato vertical. Esto será importante cuando creemos un nuevo módulo.

Seleccionamos **File>New>Module MXML** (Archivo>Nuevo>Módulo MXML) para ejecutar el nuevo asistente de módulos dentro de Flex Builder.

En el área de vista en árbol del asistente, seleccionamos el directorio de módulos que acabamos de crear o escribimos **FlexGrocer/src/modules** en la zona de entrada de texto de la carpeta ascendente.
Especificamos ModuloPedido en el campo File name (nombre de archivo) de este nuevo módulo y Layout a vertical para que se corresponda con el formato vertical de la clase Pedido actual.
Borramos los campos Width y Height ya que, como en la clase Pedido existente, especificaremos el ancho de ModuloPedido y la altura en QuickDelivery.mxml. Comprobamos que el botón de opción **"Optimize for the application"** (Optimizar tamaño para la aplicación) está seleccionado y que QuickDelivery.mxml está seleccionado en el cuadro combinado que hay a la derecha.

Seleccionamos QuickDelivery.mxml y aquí le decimos al compilador de Flex que queremos utilizar este módulo desde dentro de la aplicación QuickDelivery. El compilador intentará asegurarse de que las clases utilizadas por la aplicación principal y por este módulo no se incluyen más de una vez. Esto ayuda a reducir el tamaño de descarga del módulo.

Ahora es necesario indicarle a Flex cuándo y dónde cargar y visualizar este módulo. Tendremos que usar la etiqueta **<mx:ModuleLoader>** para cargar de forma dinámica el nuevo módulo.

En la aplicación QuickDelivery.mxml desde el directorio /src, añadimos una etiqueta <mx:ModuleLoader> donde estaba anteriormente la etiqueta <v:Checkout>, luego configuramos la propiedad ID de ModuleLoader a Pedido y configuramos Width y Height al 100%.

```
</mx:HBox>
<mx:ModuleLoader id="pedido' width="100%" height="100%"/>
</mx:ViewStack>
```

Ahora cuando agreguemos artículos al carro de compras y hacemos clic en **Pedido** observaremos una pausa momentánea antes de que se visualice la pantalla del pedido. En este momento Flex está cargando ModuloPedido.mxml por primera vez y lo está visualizando en la pantalla. El módulo sólo se carga cuando es necesario, por lo que los usuarios de la aplicación sólo cargarán ahora ModuloPedido y el código relacionado en el caso de que se seleccione la sección Pedido de la aplicación.

6.6.2 Runtime Shared Librarles

En esta sección trataremos sobre la división de las aplicaciones en partes, con la intención de compartir el código de varias aplicaciones para reducir el tamaño acumulado de los archivos de la aplicación.

Para esto necesitamos entender las aplicaciones creadas, y nos damos cuenta que tienen algunas cosas similares. Una de estas características iguales es que todas las aplicaciones usan clases de la estructura Flex, como VBox, HBox y muchas otras. QD Estadísticas y QuickDelivery incluso comparten parte del código que hemos escrito, como eventos personalizados y valueObjects. Y como vemos todo esto nos va a llevar a la duplicidad de la información.

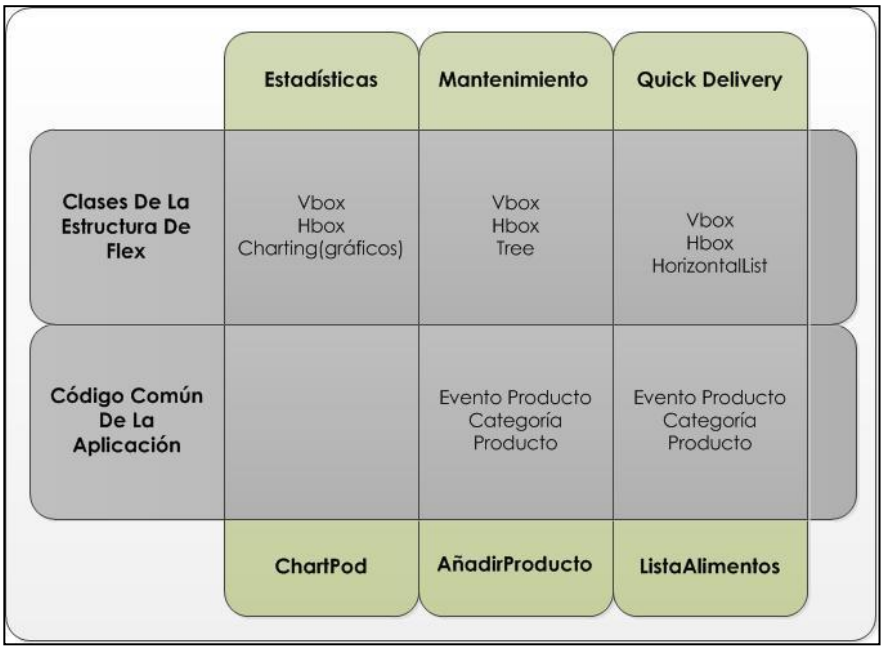


Figura 6. 58 Duplicidad de componentes de las aplicaciones Flex creadas.

Como vemos, cada aplicación es una combinación del código de la estructura de Flex, del código escrito durante el desarrollo y también del código que es único para cualquier aplicación determinada.

Runtime Shared Librarles (RSLs) nos da la posibilidad de solucionar este problema dejándonos separar el código que es común en las aplicaciones y que por lo tanto puede ser compartido por estas aplicaciones. Este código ya no se volverá a crear en la aplicación, sino que se mantiene en una biblioteca separada que se carga durante el tiempo de ejecución. Incluso RSL puede guardarse en la memoria caché del cliente por lo que no tiene que descargarse cada vez que se utiliza la aplicación.

Al utilizar RSL la estructura de la figura anterior puede ser optimizada organizándose de la siguiente manera:

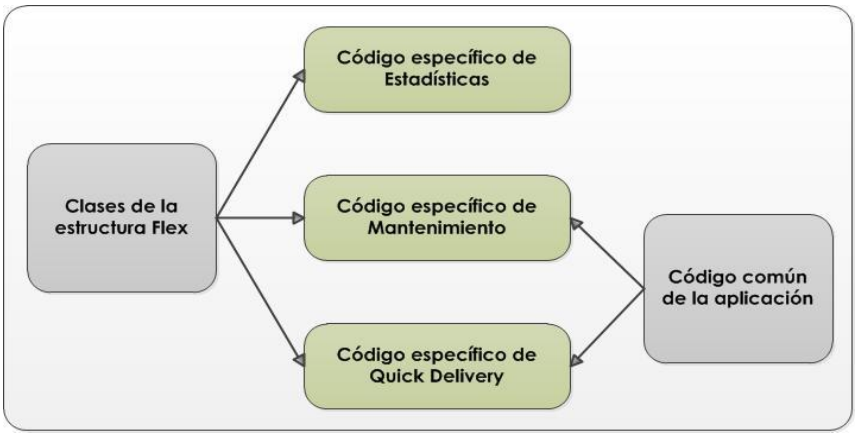


Figura 6. 59 Componentes del proyecto reorganizados con RSL

El código para la estructura de Flex y el código común de la aplicación, está en archivos separados. Esto ayuda a reducir de forma grande el tamaño de las aplicaciones QD Estadísticas, QD Mantenimiento y QuickDelivery ya que no existe duplicidad de información. Para entender este efecto a continuación mostramos los tamaños en bytes de las aplicaciones antes y después de utilizar las RSL.

Aplicación	Como aplicación independiente	Cuando comparte código utilizando RSL	Cantidad reducida
Estadísticas	417.792 KB	163.840 KB	253.952 KB
Mantenimiento	430.080 KB	167.936 KB	262.144 KB
Quick Delivery	434.176 KB	135.168 KB	299.008 KB
Total KB	1.282.048 KB	466.944 KB	815.104 KB

Figura 6. 60 Comparación de los tamaños de las aplicaciones antes y después de usar RSL

En este caso, la RSL que contiene los distintos elementos estructurales tiene unos 500 KB. Esta RSL será necesaria para ser descargada una vez y guardada en la memoria caché del cliente. Esto significa que la primera vez que el usuario acceda a la aplicación tendrá que descargar los 500 bytes dé RLS y la aplicación.

Si un usuario trabaja con las tres aplicaciones, las RSL son una gran ventaja. Incluyendo los 500 KB de descarga inicial, sólo tendrán un tamaño acumulado de descarga de aproximadamente unos 966 KB, que es significativamente menos que los 1.282 KB de las aplicaciones originales. Las descargas posteriores y el cambio entre las aplicaciones serán mucho más rápidas.

Por lo contrario, si hay usuarios que sólo usan una aplicación, por ejemplo QuickDelivery, tendrán una experiencia inicial de descarga mucho peor (500.000 + 135.168 = 635.168 bytes en contra de los 434.176 originales).

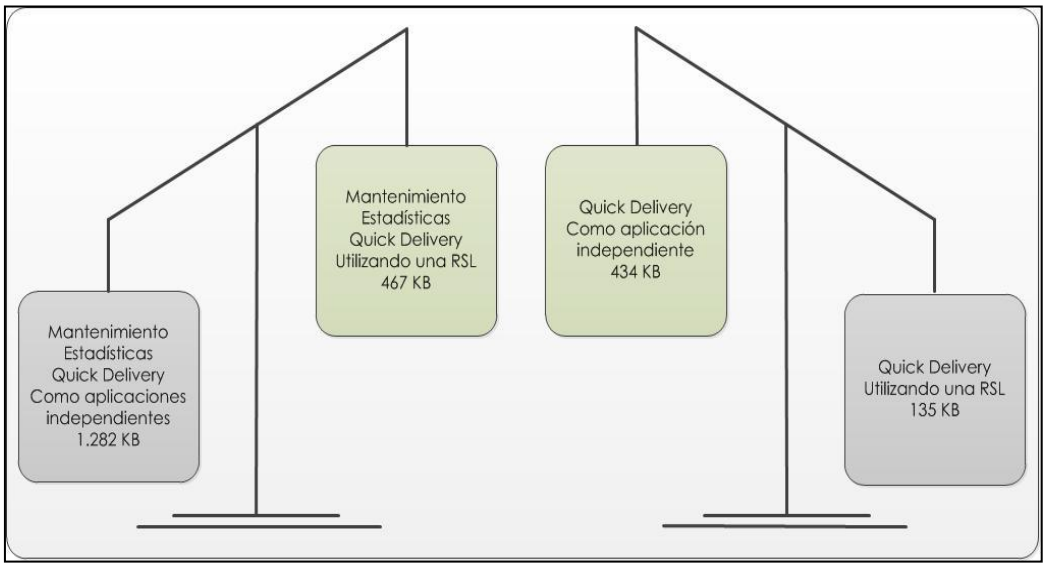


Figura 6. 61 Comparación de los tamaños de las aplicaciones

La razón de esto es que las RSL buscan reducir el tamaño acumulado de varias aplicaciones y no el tamaño de una única aplicación.

Si movemos todas las clases duplicadas a una única RSL, tendremos una RSL que contendría VBox, HBox, HorizontalList, Tree y los componentes Charting como vemos en la siguiente figura.

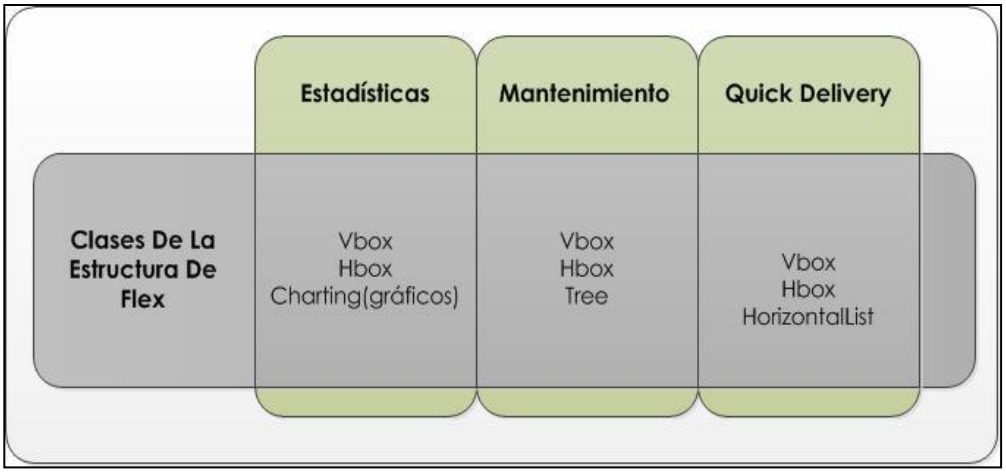


Figura 6. 62 Clases de nuestras aplicaciones Flex Quick delivery

El objetivo de almacenar la estructura propia de Flex en la memoria cache del cliente está en que se puede lograr una disminución significativa del tamaño de la aplicación separando los elementos comunes en una única *Runtime Shared Libran* (RSL), que podría ser reutilizada entre las aplicaciones.

Por ejemplo la cantidad de código duplicado descargado por todos los usuarios de cada aplicación Flex tanto en Internet como en redes privadas en todo el mundo. Cuando un usuario visita dos sitios Web distintos que utilizan Flex, normalmente se fuerza al usuario a volver a cargar las partes aplicables de la estructura.

Para resolver este problema y mejorar la experiencia del usuario para todos aquellos que utilicen una aplicación Flex, Adobe ha desarrollado una RSL de la estructura Flex que el usuario puede descargar una vez y después volver a utilizar en una futura aplicación Flex. Cuando un usuario visualiza una aplicación Flex alojada en un sitio Web, Flash Player comprueba primero una versión apropiada de la RSL estructural en la memoria caché del programa. Si se encuentra la versión correcta, se utiliza; si no se encuentra, se descarga y se almacena para un uso posterior. Cuando el usuario vuelva a visitar la aplicación Flex en otro sitio Web, Flash Player realiza la misma búsqueda y esta vez encuentra la estructura en la memoria caché.

En última instancia, la primera vez que un usuario visualiza una aplicación Flex, tendrán una descarga inicial mayor. Sin embargo, cada visita posterior a cualquier aplicación Flex será significativamente más rápida.

Una vez creada nuestra librería iremos a implementarla en nuestras aplicaciones.

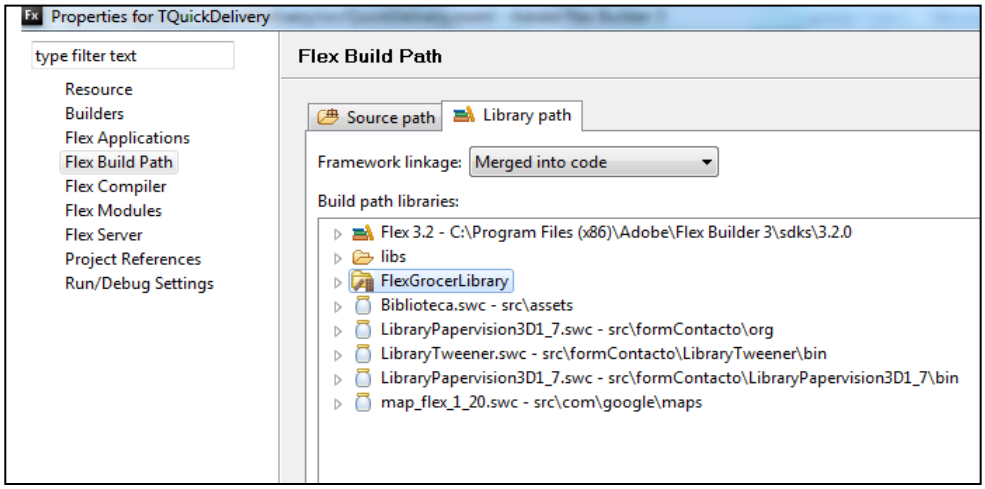


Figura 6. 63 Creación de Librerías de componentes estructurales de Flex para Quick Delivery

Nuestro proyecto Library se muestra a continuación:

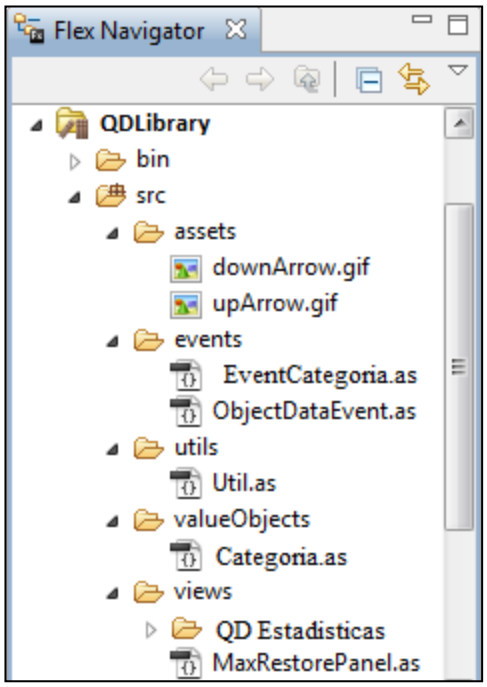


Figura 6. 64 Librería de componentes QDLibrary

6.7 Implementando la aplicación de Escritorio (Air Desktop)

Lo que haremos en esta sección del capítulo será implementar nuestro proyecto para que funcionen directamente sobre nuestras máquinas de escritorio y no solo en los navegadores como en las aplicaciones tradicionales de Internet. Como nos podemos dar cuenta el hecho de que una aplicación funcione solamente a través de un navegador genera grandes limitaciones como el que las aplicaciones no pueden interactuar directamente con el sistema de archivos, también estas aplicaciones no pueden ejecutarse como servicios en segundo plano y sabiendo que estas aplicaciones dependen casi en su totalidad de una conexión a Internet.

La solución que proponemos ante este problema es utilizar la tecnología Adobe Integrate Runtime (AIR), lo cual nos da la posibilidad de crear aplicaciones fuera del navegador Web y utilizar los recursos del computador de forma directa sin importar el SO en el que se ejecute.

Con la tecnología AIR podemos tener acceso directo al sistema de archivos, además las aplicaciones de AIR admiten las características de arrastrar y soltar elementos directamente desde el sistema operativo a la aplicación, también podemos ejecutar estas aplicaciones en segundo plano. Air nos ofrece una base de datos incrustada lo cual permite al cliente capacidades de almacenamiento mayores.
El tiempo de ejecución AIR, funciona como una máquina virtual para las aplicaciones con la extensión AIR, proporcionando una capa base entre la aplicación y el sistema operativo del usuario. Es por este tiempo de ejecución que AIR se puede instalar de forma idéntica pos los diferentes sistemas operativos.

El instalador de Adobe AIR se encuentra disponible para los sistemas operativos de Windows, Mac OS y Linux como se demuestra en su sitio web de descarga:

Figura 6. 65 Página de descarga de Adobe AIR en sus diferentes aplicaciones

6.7.1 Crear aplicaciones AIR para Nuestro Proyecto QuickDelivery

Ahora que nuestro proyecto QuickDelivery está completo, procederemos a desarrollar las aplicaciones de escritorio.

Para esto se necesita de dos archivos, uno es la aplicación MXML y el otro es el archivo application.xml que define las opciones de implementación de este frame.

Lo primero que haremos para crear nuestras aplicaciones de escritorio será crear un nuevo proyecto, al que lo llamaremos QuickDeliveryDesktop, pero en lugar de configurar el Application Tipe como Web Application lo configuraremos como Desktop Application y como Application Server seleccionamos None.

Haremos clic en finalizar para continuar con la construcción de nuestra aplicación de escritorio.

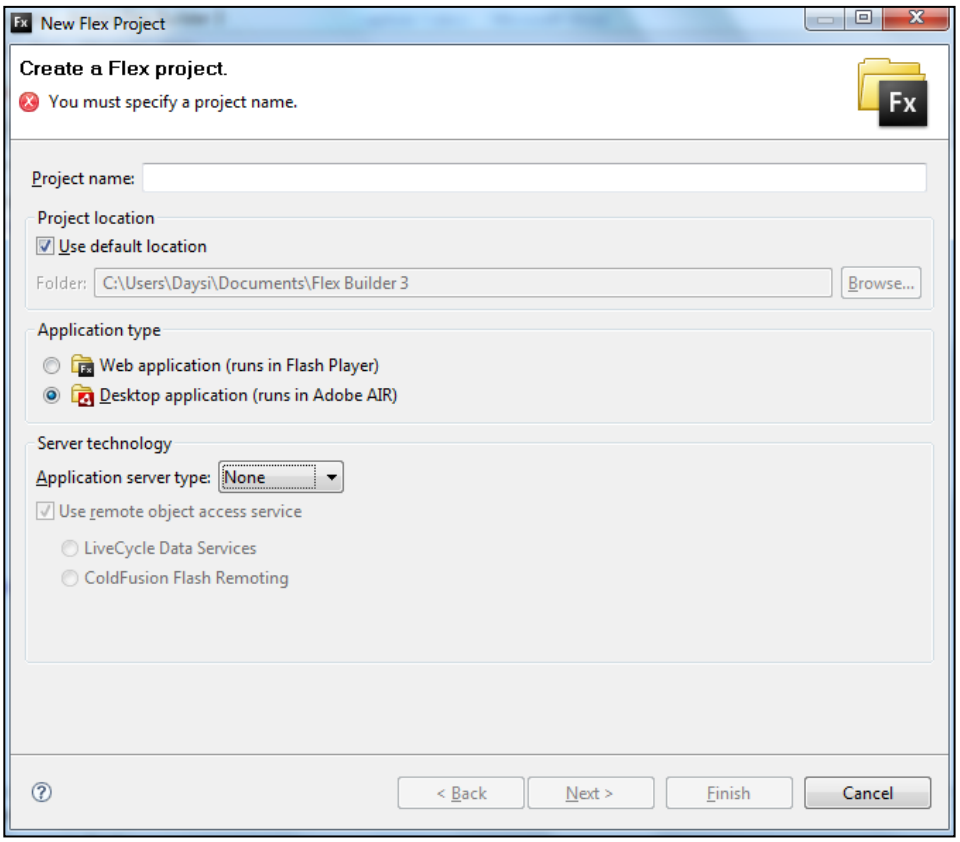


Figura 6. 66 Opciones de la creación del proyecto de Escritorio

Damos clic en siguiente y en la pantalla de configuración final del proyecto que estamos creando en el campo Application Id pondremos el nombre com.QuickDelivery.dektop (cabe recalcar que en las aplicaciones de escritorio cada aplicación debe de tener un ID único), esta ID se utilizará para registrar la aplicación con el sistema operativo, por lo que si se intenta instalar una aplicación AIR que ya se ha instalado se le preguntara al usuario si quiere desinstalarla, o volverla a instalar.

Ahora usando la librería que anteriormente hemos creado añadiremos estas funcionalidades a nuestra nueva aplicación:

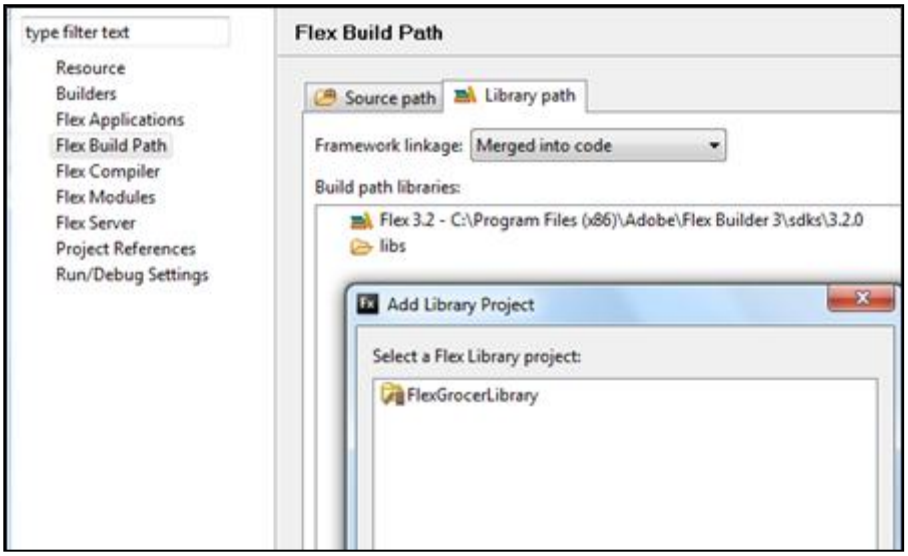


Figura 6. 67 Función añadir Librería de componentes a nuestro sistema

Usaremos el Archivo que ya hemos creado anteriormente QuickDeliver.mxml y copiaremos todo su contenido en nuestro proyecto de escritorio y procederemos a cambiar las etiquetas <mx:Application> con <mx:WindowedApplication>. Procederemos a ejecutar el proyecto y lo probaremos. Del mismo modo como hemos hecho con la aplicación QuickDelivery, crearemos la versión de escritorio de las aplicaciones QD Estadísticas y QD Mantenimientos.

Figura 6. 68 Aplicación QuickDelivery Desktop creado con tecnología AIR

Figura 6. 69 Aplicación QD Estadísticas Desktop creado con tecnología AIR

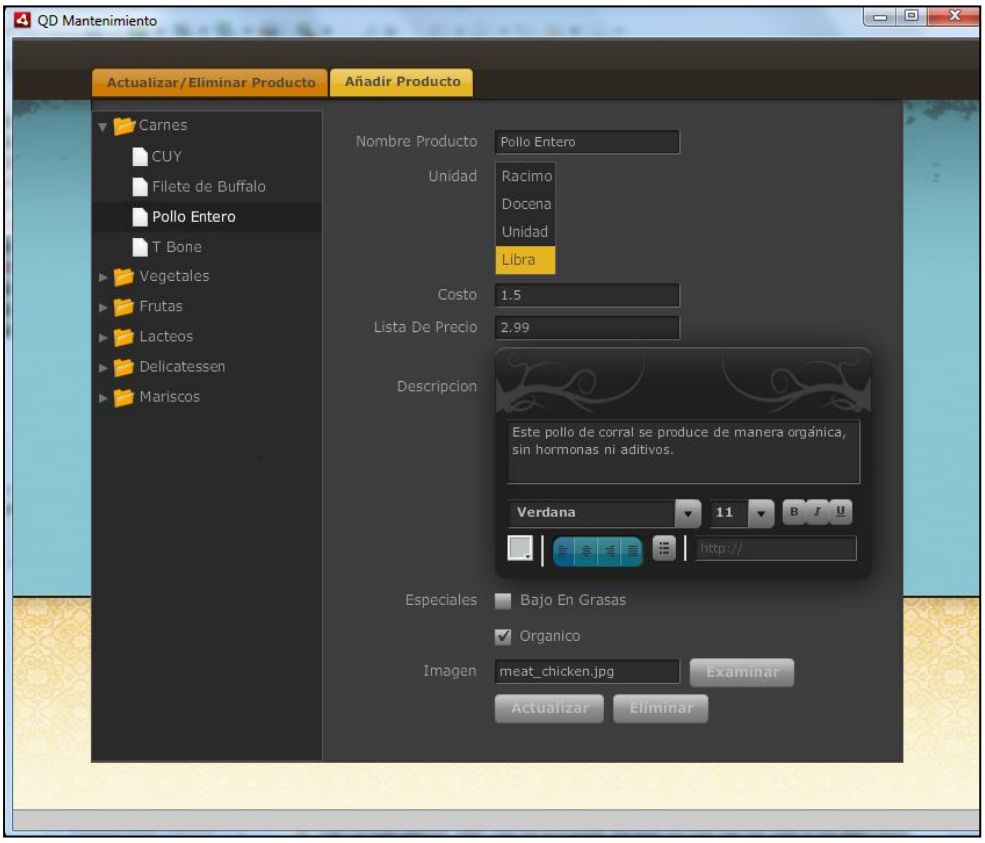


Figura 6. 70 Aplicación QD Mantenimiento Desktop creado con tecnología AIR

Anexo 2 (Instalación Adobe Air)

6.7.2 Exportar el proyecto AIR

Para poder exportar nuestros proyectos realizaremos los siguientes pasos:

- Seleccionamos el proyecto QD Mantenimiento y vamos a Project > Exprot Release Builder.

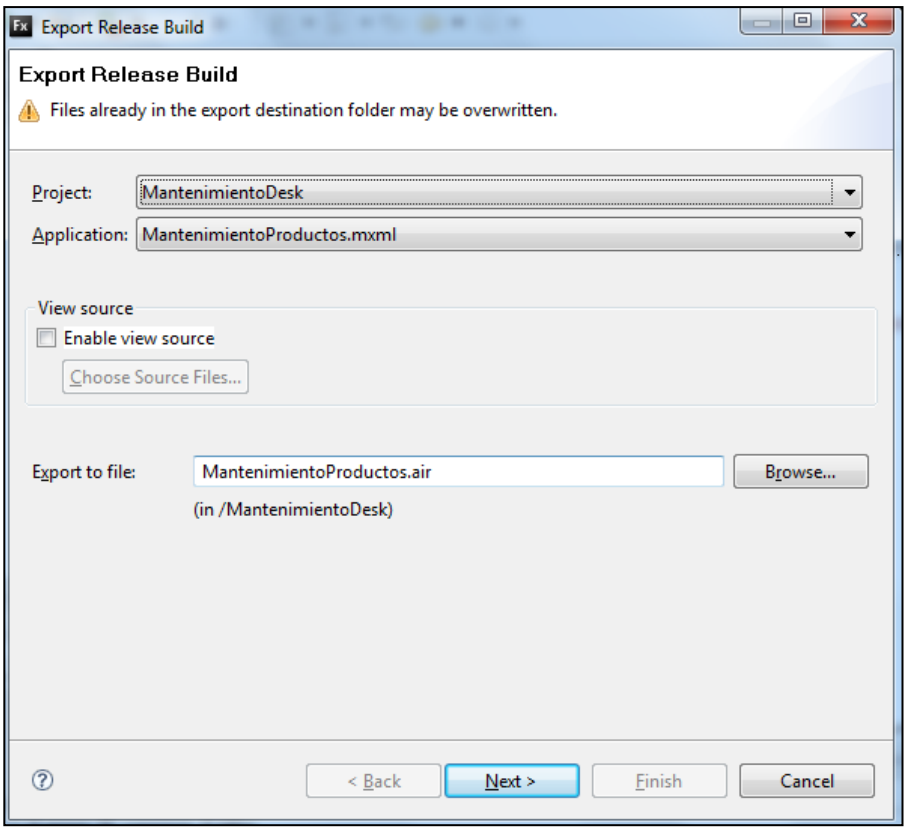


Figura 6. 71 Configuración de Proyecto a Exportar

- En el siguiente paso requerimos indicar un certificado de seguridad con el cual logramos imprimir un tipo de firma digital, para esto hay varias opciones:

  - ✓ Certificado digital Verising o Thawte.
  - ✓ Se puede crear y utilizar un certificado digital autofirmado.
  - ✓ Se puede empaquetar una aplicación sin firma.

  En nuestro caso seleccionaremos una aplicación autofirmada.

- Creamos nuestro certificado con la siguiente información:

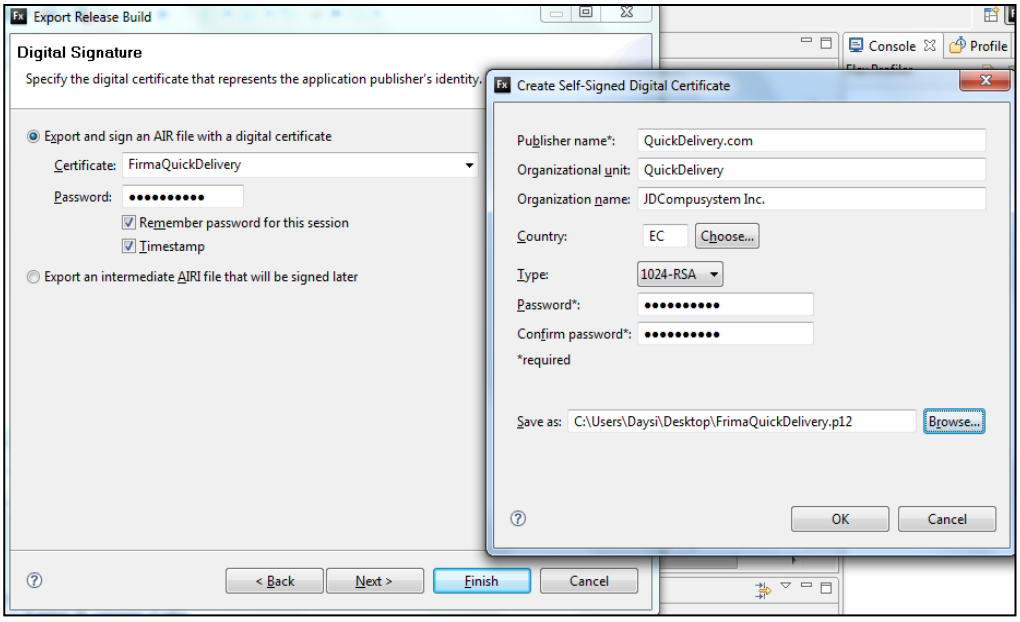


Figura 6. 72 Creación de Certificado Digital autofirmado

- Damos clic en siguiente y la pantalla que nos muestra Flex nos permite especificar los archivos que se incluirán en el archivo exportado. Confirmamos esta información y damos clic en Finish.

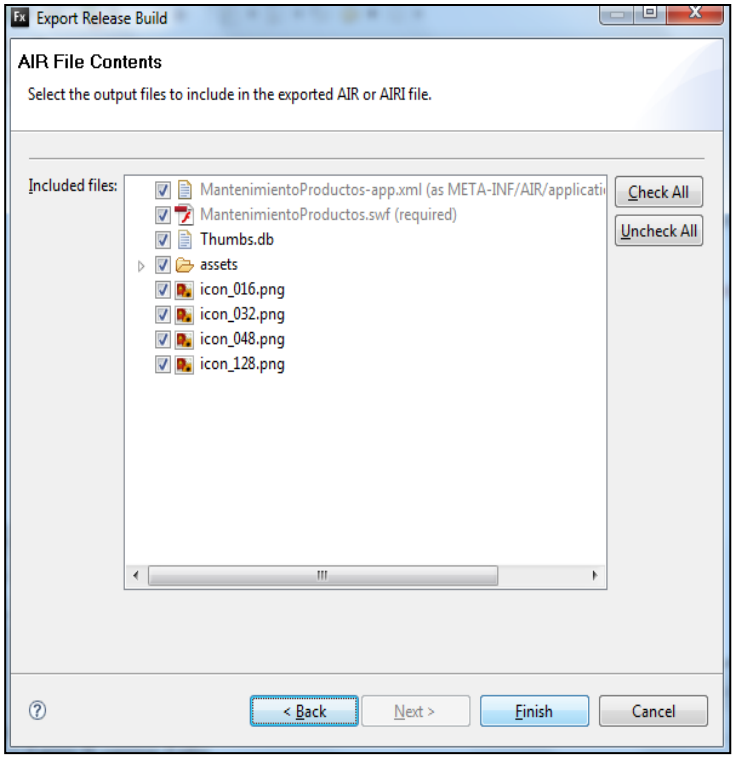


Figura 6. 73 Seleccionamos archivos que vamos a exportar

- Cuando hemos terminado este procedimiento buscamos el archivo generado, el mismo que llevará el nombre de Mantenimiento.air.

Figura 6. 74 Archivos Generados

- Para instalar nuestra aplicación solamente tenemos que ejecutarla y seguir los pasos que nos detalla en pantalla.

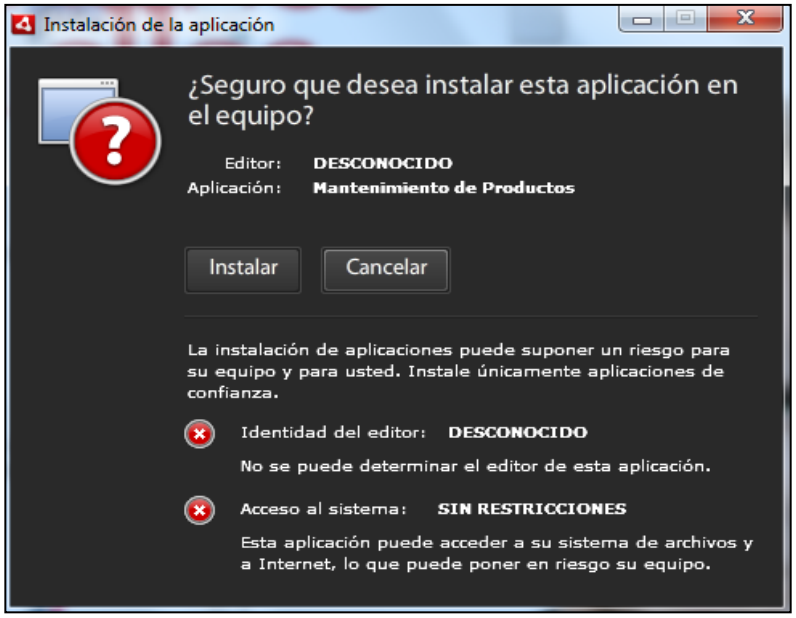


Figura 6. 75 Pantalla de Instalación programa Mantenimiento de Productos

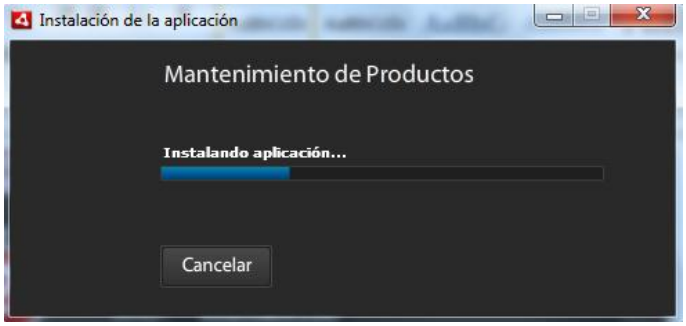


Figura 6. 76 Instalación en curso Mantenimiento de Productos

- Mostramos la aplicación instalada en nuestro sistema operativo.

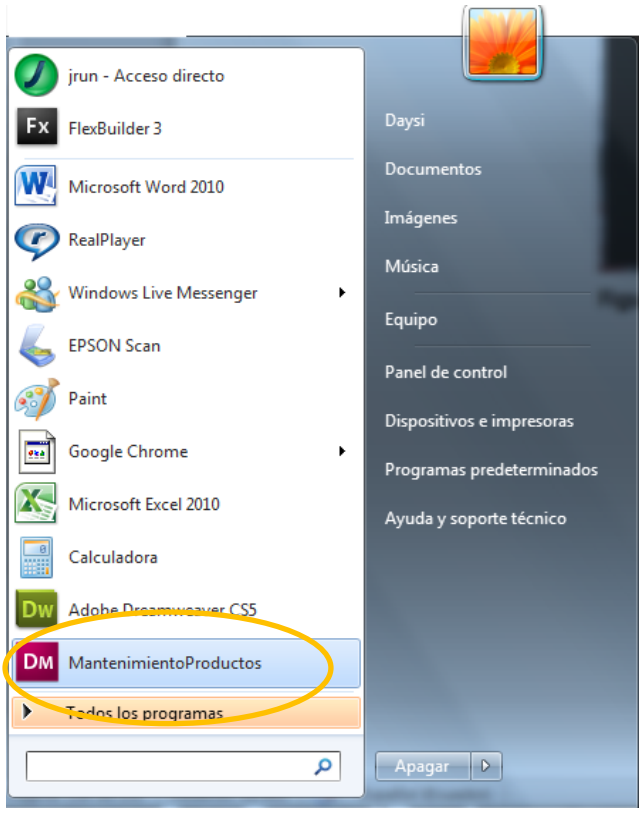


Figura 6. 77 Programa Mantenimiento de productos anclado en el menú de inicio de Windows

6.8 PaperVision 3D

Antes de iniciar con la teoría y puesta en marcha de las librerías PaperVision indicaremos que para fines de aprendizaje vamos a desarrollar un formulario de contacto de clientes, cuya característica principal será que tiene propiedades visuales, movimientos en un plano en 3 dimensiones, y se puede girar en el espacio, además este formulario tendrá conexión con una base de datos MySql en la cual guardara la información de contacto de los clientes.

PaperVision 3D es una biblioteca de clases creadas para ActionScript, Adobe Flash y Flex . Con el aumento de popularidad de esta biblioteca entre los desarrolladores de todo el mundo, PaperVision se ve en la necesidad de disponer en Google su código y hacerlo de carácter abierto.

El objetivo principal del proyecto, radica en la ayuda al programador para desarrollar modelos en 3D.

A pesar de la liberación de Flash en su versión 10, aun su motor 3D nativo es muy simple y sin la suficiente versatilidad en el desarrollo de modelos 3D. PaperVision y otras bibliotecas existentes se siguen utilizando y mejoran con el transcurso del tiempo, esto gracias a las actualizaciones de la máquina virtual ActionScript.

Mientras por un lado la tecnología PaperVision da grandes oportunidades, uno de los grandes inconvenientes de este proyecto es que requiere enormes cantidades de memoria, pudiendo causar problemas de rendimiento.

Esto se debe principalmente al hecho de que la ejecución de PaperVision se delega por completo a la usuario, que hace poco uso de la GPU, basándose casi exclusivamente en la potencia de cálculo del procesador.

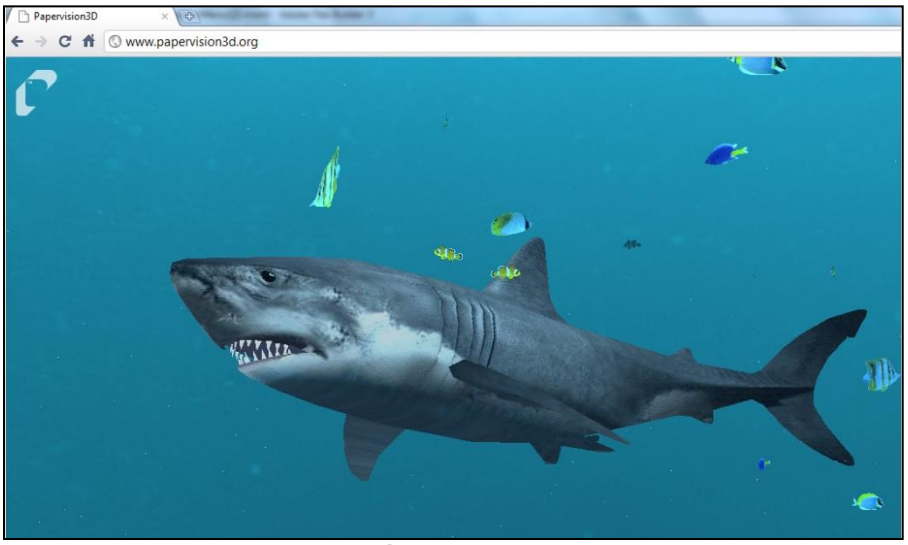


Figura 6. 78 Página Oficial PaperVision 3D

Antes de empezar con el desarrollo vamos a hacer una pequeña introducción para entender cómo funciona y como poner en marcha estas librerías, además llegar a comprender conceptos como "viewport", "escena", "renderizado", entre algunos temas más.

Papervision3D no son más que librerías de clases de ActionScript que no requieren ningún tipo de instalación, y de las cuales nuestras aplicaciones harán uso para implementar sus funciones.

6.8.1 Descargando Las librería PaperVision 3D

Las librerías de PaperVision 3D se localizan en repositorios o bodegas de control de versiones, y por su gran demanda por parte de muchos desarrolladores, sus creadores, han decidido aliarse con Google Code para facilitar su difusión, lugar donde se encuentra almacenado este repositorio.

http://papervision3d.googlecode.com/svn/trunk/as3/trunk/docs/index.html

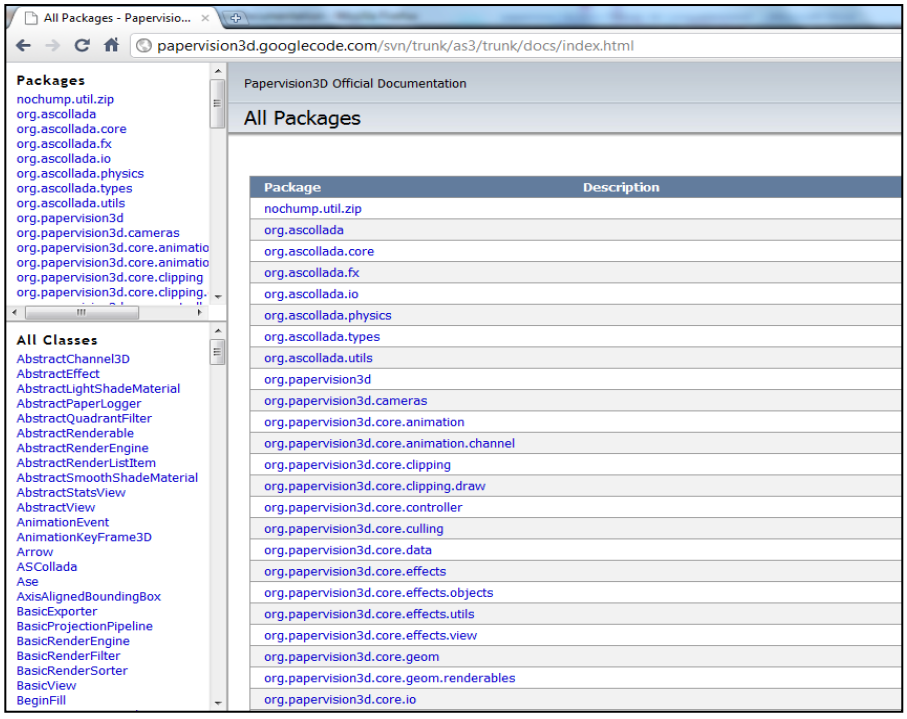


Figura 6. 79 Repositorio de PaperVision almacenado en las bodegas de Google

Si queremos descargar cualquier clase de este repositorio necesitamos un cliente de descarga de subversiones (Google solo trabaja de esta manera), y el que nosotros usamos se llama Tortoise, el cual se integra con el explorador de Windows. Seleccionamos el link a descargar, en nuestro caso la librería check out, esta librería contiene cuatro carpetas de las cuales usaremos "branches/ GreatWhite/ src/". Pondremos todas las librerías en una sola carpeta donde tendremos todas las clases.

Ahora también podemos descargarnos las librerías LibraryPaperVision3D1_7.swc y LibraryTweener.swc, las cuales podemos encontrar en: http://code.google.com/p/tweener/.

La librería Tweener es una clase para crear tweenings (ir de un fotograma clave a otro, guiado en una ruta de acceso determinada o marco) y otras transiciones usando código ActionScript para proyectos basados en la plataforma Flash.

Tweener nos va a ayudar a mover los objetos de la pantalla utilizando sólo el código de programación, en substitución de la línea de tiempo de Flash. La idea de una clase de interpolación creada por código, es que la animación sea más fácil de mantener y controlar.

Lo siguiente será agregar estas librerías a nuestro proyecto en Flex Builder 3, lo cual lo haremos haciendo clic derecho sobre nuestro proyecto QuickDelivery, seleccionamos Properties y a continuación escogemos Flex Build Path, y de este seleccionaremos la pestaña Library Path y pulsaremos en el botón Add Folder de donde seleccionamos nuestra carpeta en la que hemos guardado todas nuestra librerías PaperVision.

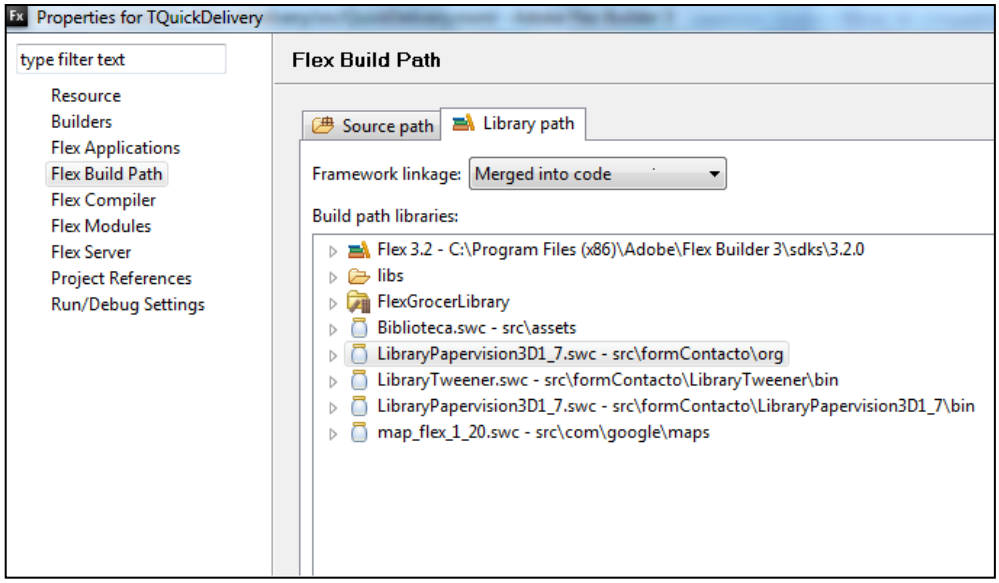


Figura 6. 80 Añadiendo Librerías PaperVision al proyecto QuickDelivery

Ahora trataremos de explicar un poco sobre los términos básico que usaremos en el desarrollo de nuestro formulario 3D

Viewport: Esto es la parte visible de nuestros elementos 3D. Es mediante el cual podemos configurar el tipo de renderizado, la cámara y la escena. Si definimos un viewport de 500 x 400 píxeles, todo lo que se encuentre fuera de esas dimensiones no se verá.

Escena (Scene): Se refiere al contenedor de todos los elementos visibles y formas que tendrá nuestra aplicación. Podemos decir que es como un "display list" interno de PaperVision.

Motor de renderizado (Renderer): Es mediante el cual renderizamos la escena. La "renderización" es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D. En PaperVision, cada vez que modificamos algo en la escena 3D, debemos renderizar nuevamente. Lo que se suele hacer es un método que se encargue de ello y que se ejecuta en un intervalo fijo mediante el evento "ENTER_FRAME".

Cámara: en PaperVision podemos controlar la posición, rotación y zoom de la cámara que visualiza el contenido.

Para trabajar con estos elementos es necesario realizar la configuración inicial de los mismos:

```
<mx:Script>
        <![CDATA[
                import mx.controls.Button;
                import  formContacto.jel.com.pv3d.Escenario3D;

                private var escenarioInteractivo:Escenario3D
                private function main():void {

                        var botonReset:Button = new Button();
                        botonReset.width = 150;
                        botonReset.height = 50;
                        botonReset.label = "Limpiar Formulario";
                        botonReset.y = 170;
                        botonReset.x = 665;
                        botonReset.alpha = 0;
botonReset.addEventListener(MouseEvent.MOUSE_UP, MouseUp);
                        this.addChild(botonReset);

                escenarioInteractivo = new Escenario3D(540, 380);

                        this.addChild(escenarioInteractivo);

                }
                private function MouseUp(evt:MouseEvent):void {
                        escenarioInteractivo.reset();
                }
        ]]>
</mx:Script>
```

Con el código que hemos escrito estamos inicializando nuestro formulario 3D y con el método:

```
import  formContacto.jel.com.pv3d.Escenario3D;
```

Llamamos a las clases y objetos del Script que crea el entorno de este objeto y sus propiedades.

Ahora añadir un elemento de tipo biblioteca de Flash Player llamado Biblioteca.fla que será nuestra escena y vamos a incluir una ecuación para controlar el movimiento de cámara, de tal forma que podremos interactuar con los movimiento del mouse.

```
private function onEnterFrame(evt:Event):void {
        var posX:Number;
        var posY:Number;

        if (giraVisor) {

        posX = contenedor_spt.mouseX;
        posY = contenedor_spt.mouseY;

        plano.rotationX = incRotacionX + 0.5*(posY-posMouseY);
        plano.rotationY = incRotacionY - 0.5*(posX-posMouseX);
        }

        escenario3D.renderCamera(camara);
  }
```

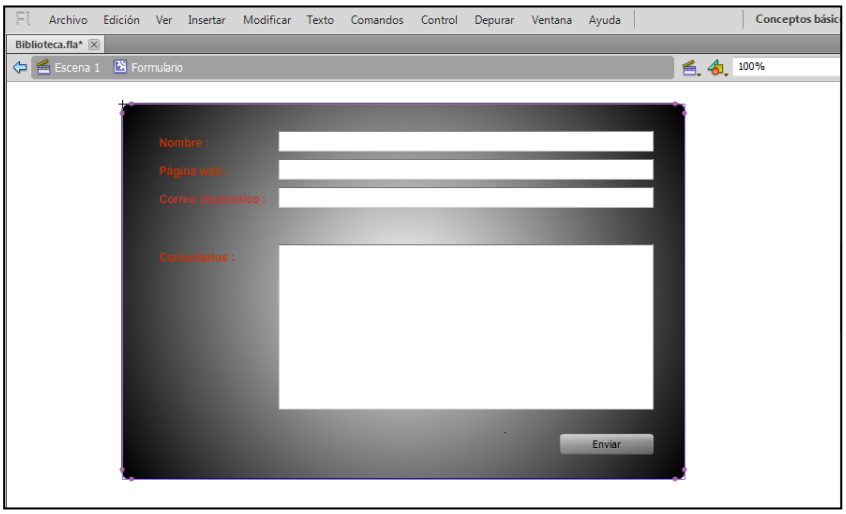


Figura 6. 81 Creación del componente de la escena en Flash CS5

Para finalizar crearemos el código que nos permitirá interactuar con el movimiento de nuestra escena.

```
var radio:uint = 2000;

var angulo:Number = mouseX + mouseX + stage.stageWidth / 2;

camera.x = radio * Math.cos(angulo / 100);
camera.z = radio * Math.sin(angulo / 100);
camera.y = - mouseY + stage.stageHeight / 2;

Render();
}
```

El resultado visual de nuestro formulario será el siguiente:

Figura 6. 82 Pantalla Formulario de Contacto de Cliente 3D

Como podemos observar nuestro formulario tiene movilidad en el plano y proyección en las 3 Dimensiones, lo que le da un efecto muy llamativo. Lo que pretendemos lograr con este ejemplo, es demostrar la versatilidad y potencial de trabajar con tecnología RIA a la hora de desarrollar nuestros sitios Web y también nuestras aplicaciones de escritorio, pudiendo por este medio no solo llegar a este tipo de plataformas, (tanto la Web, como escritorio), sino, proyectarnos al futuro inmediato que son los teléfonos móviles, dispositivos tipo Tablet, Televisores con Internet, entre algunas tecnologías que podemos mencionar.

El siguiente paso en este ejemplo será mostrar la conexión que estamos realizando entre este objeto y nuestra base de datos MySQL. Haremos este ejemplo utilizando esta tecnología para demostrar la Flexibilidad

de nuestro proyecto al momento de conectarse con diferentes servidores y distintas bases de datos.

Lo primero que mostraremos es como nos conectamos con el modelo diseñado en Flash CS5, la biblioteca de la cual obtenemos los diferentes componentes visuales.

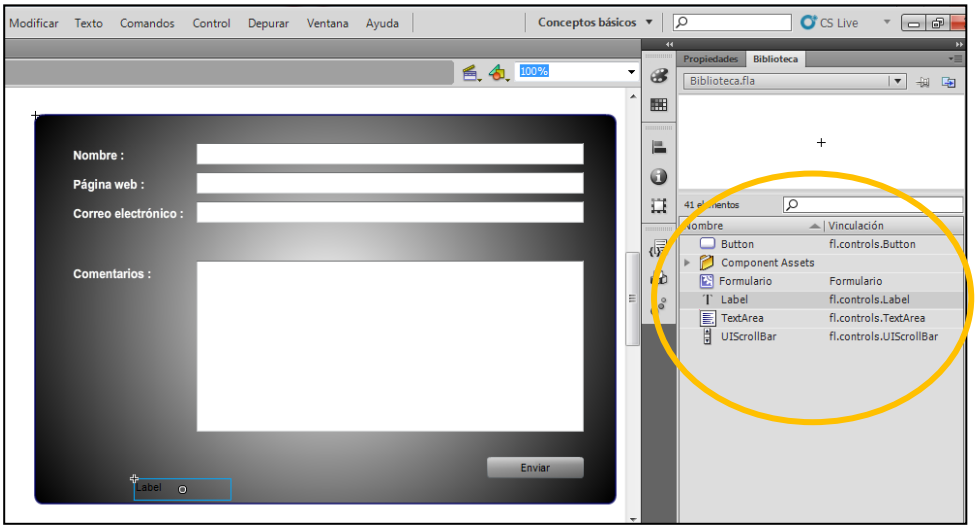


Figura 6. 83 Creación de los componentes usados en nuestro Formulario 3D, usando Flash Professional CS5

Ahora en nuestra clase Escenario3D.as vamos a crear el código necesario para poder acceder a contenido de estos Objetos haciendo un llamado a estas clases.

```
import caurina.transitions.Tweener;

import flash.display.BlendMode;
import flash.display.MovieClip;
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;

import formContacto.jel.com.main.CanvasFlash;


private function handleBTNClick(evt:MouseEvent):void {

Tweener.addTween(plano, {x:(1000*(0.5-Math.random())), y:(1000*(0.5-
Math.random())), z:1000, scale:0, time:2.0, transition:"easeInOutQuad"});
```

```
                var campoNom:String;
                var campoDirec:String;
                var campoProfe:String;
                var campoText:String;

    //esto es para tomar los valores del Biblioteca.swc del form3D
                campoNom = material.movie["campoNombre"].text;
                campoDirec = material.movie["campoDireccion"].text;
                campoProfe = material.movie["campoProfesion"].text;

                campoText = material.movie["campoTexto"].text;
send_data(campoNom, campoDirec, campoProfe, campoText); }
```

Lo siguiente que haremos es importar las clases necesarias para la creación de nuestra conexión con MySQL.

```
                import mx.rpc.events.ResultEvent;
                import mx.rpc.http.HTTPService;
```

Ahora escribiremos la lógica de programación necesaria para crear nuestro HTTPService con el que enviaremos los valores obtenidos del objeto Biblioteca.swc anuestra base de datos de MySQL.

```
private var servicios:HTTPService
[Bindable]
private var usuarios : ArrayCollection = new ArrayCollection();
[Bindable]
public var cards: Array = ["PUT","POST","GET","DELETE"];
[Bindable]

 public var selectedItem:Object = "GET";
 private var parametro:Object = new Object();
public function end_data(campoNom:String,campoDirec:String,
campoProfe:String, campoText:String) : void
        {
                //var obj : Object = new Object();
                                        useHttpService(campoNom,
        campoDirec, campoProfe, campoText);
        }
```

En esta parte hacemos la llamada al archivo conectarForm3D.php, el cual nos conectará con el servidor PHP y con la base de datos. Además es este archivo que contiene los parámetros a enviar.

```
public function useHttpService(campoNom:Object, campoDirec:String,
campoProfe:String, campoText:String):void {
servicios = new HTTPService();
servicios.url = "http://localhost/phpTesis/conecForm3D.php";
 servicios.method = "POST";
servicios.addEventListener("result", httpResult);
servicios.addEventListener("fault", httpFault);
Parametro = {"campoNom": campoNom, "campoDirec":campoDirec,
"campoProfe":campoProfe, "campoText":campoText};
servicios.send(parametro);

}
```

El archivo de conexión se detalla a continuación:

Archivo conecForm3D.php.

```
1 / 2 ▸  No hay errores de sintaxis.
 1  <?php
 2      if (!($conexion=mysql_connect('localhost','root',''))){
 3            echo "Error conectando a la base de datos.";
 4            exit();
 5      }
 6      if (!mysql_select_db("tquickdelivery",$conexion)) {
 7            echo "Error seleccionando la base de datos.";
 8            exit();
 9      }
10      function quote_smart($value)
11      {
12          if (get_magic_quotes_gpc()) {
13          $value = stripslashes($value);
14      }
15          if (!is_numeric($value)) {
16          $value = "'" . mysql_real_escape_string($value) . "'";
17      }
18      return $value;
19      }
20      if(  $_POST['campoNom'] and $_POST['campoDirec'] and $_POST['campoProfe'] and $_POST['campoText'])
21      {
22          //add the user
23          echo "entroooooooooooooooooooooooooo";
24      $fecha_actual= date("YmdHis");
```

Figura 6. 84 Archivo de conexión entre Flex 3, PHP y la base de datos de MySQL

Ahora podemos probar la aplicación y verificar que nuestros datos están debidamente guardados en la base de datos.

6.9 Uso de API´s en el proyecto QuickDelivery

Una de las extraordinarias ventajas de trabajar con tecnología RIA a la hora de desarrollar nuestras aplicaciones, es la adaptación que tenemos con los diferentes sistemas y recursos de código. Esto podemos demostrarlo usando las API´s a la hora de dar características especiales a nuestro sistema.

API es una interfaz de programación de aplicaciones traducido del inglés *Application Programming Interface*. Corresponde a un conjunto de procedimientos, funciones y métodos basados en la programación orientada a objetos, y que nos permite el uso anexado de bibliotecas de código para utilizarlo por diferente software como una capa de abstracción.

El propósito principal de trabajar con API´s es que estas nos proporcionan un conjunto de funciones de uso general, por ejemplo, Google Translate o Mapas de Google. Y es gracias a esto que los desarrolladores de software podemos usar estas librerías creadas por terceros, para no tener que escribir todo este código desde un principio, sino tan solo adaptarlo a nuestras necesidades.

En nuestro proyecto usaremos las API´s de Google como son Google Maps y Google Translate. Nuestro objetivo no viene a ser el desarrollo en concreto de una tecnología basa en estas librerías, sino más bien, el demostrar con pequeños ejemplos la adaptación y funcionalidad que podemos tener a la hora de desarrollar RIA´s.

> "Google Maps es el nombre de un servicio gratuito de Google. Es un servidor de aplicaciones de mapas en la Web. Ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo entero e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle Street View." (http://es.wikipedia.org/wiki/Google_Maps)

> "Google Translate o Traductor Google es un sistema de traducción automática gratuito proporcionado por Google Inc, basado en datos

estadísticos para traducir texto, documentos y páginas web a otras lenguas. Google introdujo su propio software de traducción en 2007."(http://es.wikipedia.org/wiki/Google_Translate)

Una vez que hemos explicado la parte teórica de este tema procederemos a desarrollar las aplicaciones que servirán de ejemplo.

La aplicación que desarrollaremos será un utilitario creado con el API de Google Maps con el que podremos visualizar mapas y ubicaciones geográficas.

Para poder comenzar a trabajar con esta tecnología debemos de tener instalado el API de Google Maps para nuestro Framework de desarrollo de software de Flex Builder 3 y además haber obtenido una clave de API de Google Maps.

El API que usaremos es Google Maps API for Flash, este se encuentra en la biblioteca como un archivo *.swc situado en el directorio lib del kit de desarrollo de Google Maps API for Flash el cual lo podemos descargar de: http://maps.googleapis.com/maps/flash/release/sdk.zip.

Este archivos SWC contienen interfaces para todas las clases públicas del entorno de desarrollo de Google Maps. Para la instalación de la clave es necesario obtener una cuenta en GMAIL, y solicitar un archivo de configuración de uso de esta aplicación en la Web de Google, en nuestro caso usaremos un archivo de configuración especial denominado "Declaración MXML" (debido a que usamos Adobe Flex 3). Esta API se compila en un archivo SWF y debe de corresponder al dominio en el que se registró nuestra licencia.

El siguiente paso será vincular la biblioteca de Google Maps API for Flash a nuestro proyecto usando el asistente que encontramos en las propiedades del proyecto.

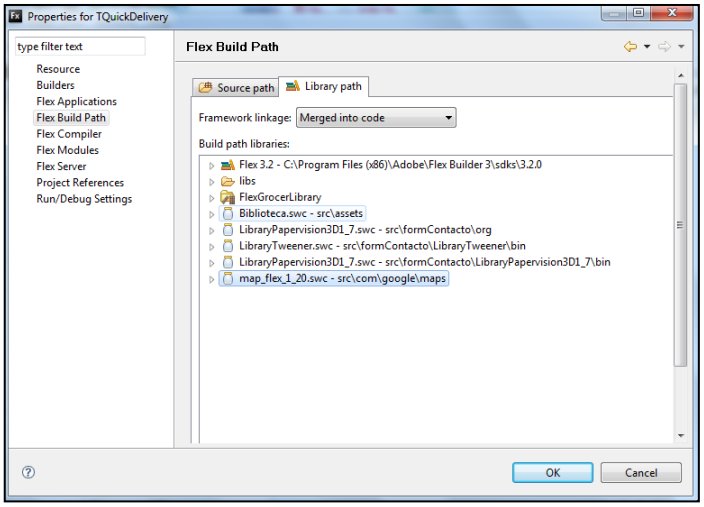


Figura 6. 85 Inclusión de la librería de Google a nuestro proyecto QD

Ahora en una nueva aplicación MXML creamos el código para llamar a las clases de Google Maps y poder usar sus funciones.

ANEXO 3 (código Google Maps)

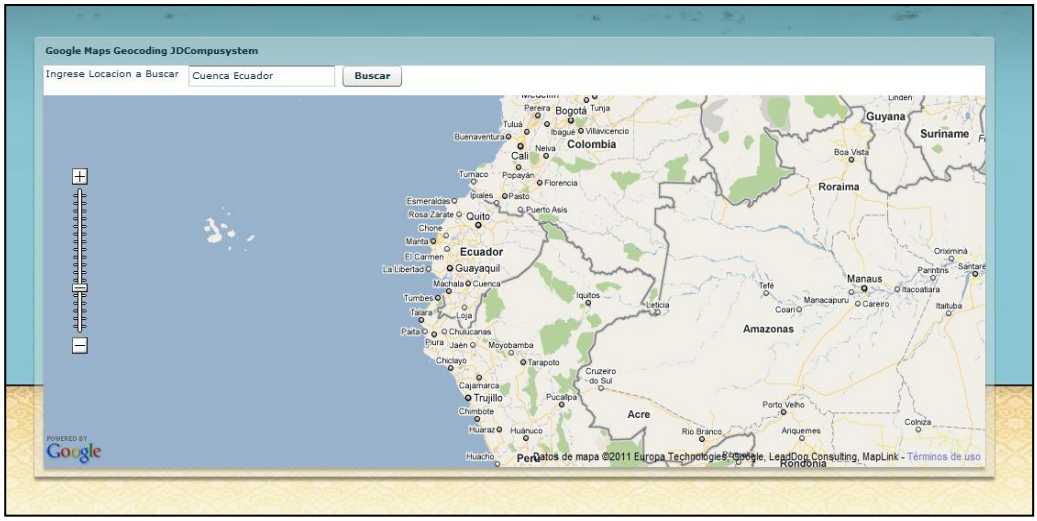


Figura 6. 86 Aplicación Google Maps Geocoding QuickDelivery

Como ya mencionamos anteriormente a continuación presentamos un ejemplo de la aplicación API de Google Translate en el proyecto QuickDelivery.

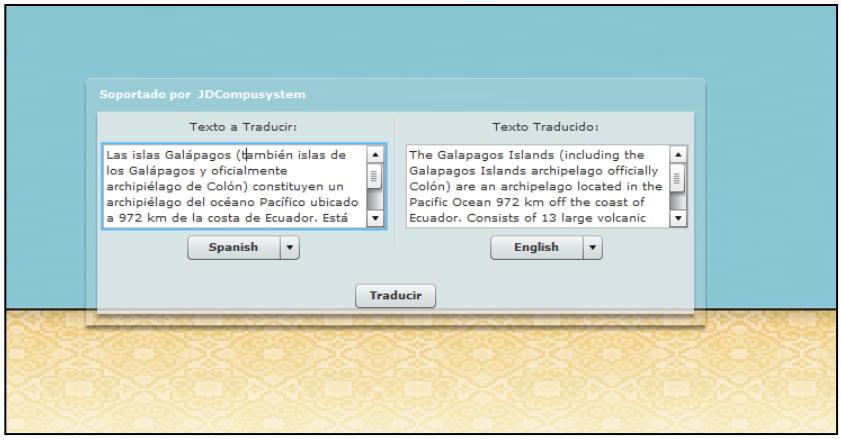


Figura 6. 87 Aplicación de Traductor usando el API de Google Translate

6.10. Conclusión

Mediante el desarrollo de este sistema hemos podido aprender de manera práctica la codificación y elaboración de un proyecto RIA utilizado herramientas y técnicas que nos proporciona esta tecnología. Como se ha mostrado la aplicación de nuevos métodos y paradigmas de programación nos abre la puerta al desarrollo de sistemas más interactivos y menos restrictivos en cuanto a la plataforma que los soportan, permitiéndonos hacer uso de los recursos propios del computador del cliente a la hora de procesar componentes complejos y pesados asignando la tarea de este procesamiento a la CPU del Cliente y no al servidor que aloja este servicio.

Además hemos mostrado la versatilidad de este sistema, al acoplar en su funcionamiento módulos y código de aplicaciones de terceros como es el trabajar con las API de Google y las librerías de PaperVision con las cuales hemos creado componentes como Formularios de contacto en 3D, además traductores que usan el motor de Google para traducir texto en múltiples idiomas y también el uso de Georeferenciación usando el sistema de mapas de Google Maps. Además hemos tratado de demostrar nuestras razones por la que hemos escogido esta temática para la elaboración de nuestro trabajo de grado.

CAPÍTULO VII

CAPÍTULO 7
PRUEBAS DE LA APLICACIÓN WEB

7.1. Introducción

Como parte del desarrollo de software orientado a aplicaciones RIA, desarrollaremos las pruebas dentro de la aplicación Quick Delivery desarrollada en Adobe Flex Builder 3 y realizaremos un reporte sobre las herramientas y técnicas utilizadas para identificar los posibles problemas que pueden hacer que esta aplicación no funcione de forma inmediata, también identificar los factores que repercuten para que esta aplicación se ejecute más lentamente o hacen que utilice más cantidad de recursos de la máquina como memoria y procesador o que simplemente no funcione correctamente. Para esta prueba usaremos herramientas y técnicas proporcionadas por Profiler de Flex Builder 3.

7.2 Análisis y pruebas de la aplicación RIA utilizando Adobe FLEX

Se llevará a cabo el análisis de la memoria y del rendimiento, dos técnicas suministrada por la nueva característica Profiler de Flex Builder 3 Pro. Para entender la necesidad de esta herramienta tendremos que aprender algunos detalles de cómo ejecuta Flash Player el código en Flex y cómo asigna y libera memoria.

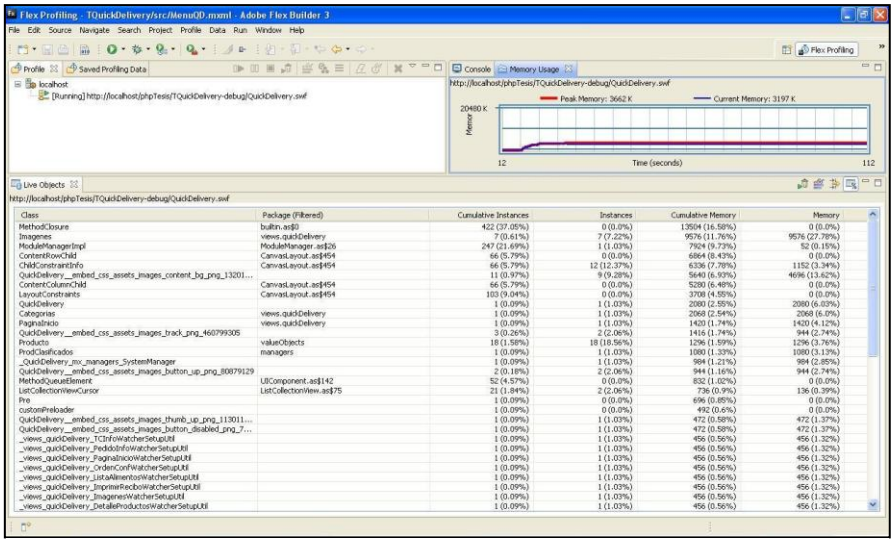


Figura 7.1 Pantalla de la utilización de la vista Live Objects en una aplicación en ejecución

7.2.1. Utilización de memoria en Flash Player

7.2.1.1. Asignación de memoria en Flash Player

Flash Player es el encargado de proveer memoria para la aplicación Flex durante el tiempo de ejecución. Cuando ejecutamos una línea de código que crea una nueva instancia de DataGrid, Flash Player suministra una parte de memoria para que la pueda ocupar esa instancia. Entonces Flash Player tiene que

pedir al sistema operativo de su ordenador memoria para poder utilizarla, con este fin. El procedimiento para pedir memoria al sistema operativo es lento, por lo que Flash Player tiene que pedir bloques mucho mayores de lo que necesita y mantiene la disponibilidad adicional para la próxima vez que el desarrollador solicite más espacio. Además, Flash Player busca memoria que ya no esté en uso, por lo que esta puede ser reutilizada antes de pedir más al sistema operativo.

7.2.1.2. Transmitir referencia o valor

Existen dos tipos de grupos principales de datos que hay que entender al momento de tratar con la memoria en Flash Player. El primer grupo se llama primitivas, e incluye Number, String ,Boolean, int, uint. Estos tipos son los responsables del transporte del valor durante las llamadas de asignación o de función. Por ejemplo si ejecutamos el siguiente código:

```
Var a: Number;

Var b: Number;

a = 5;

b = miEdad;

a = 7;
```

Se crearían dos números y asignarían cada una sus valores de forma separada. Desde un nivel muy alto, la memoria de Flash Player sería como se ve en la siguiente figura 7.2:

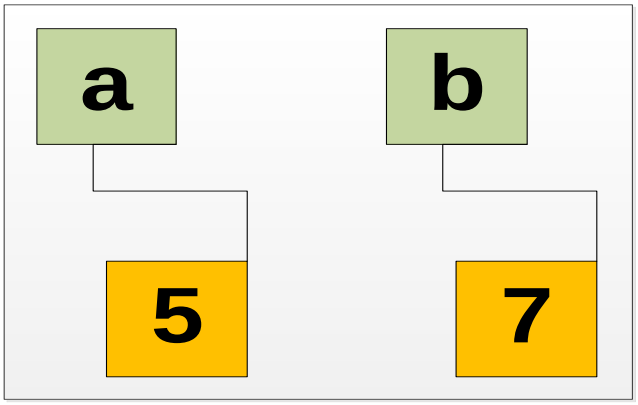


Figura 7. 2 Funcionamiento De La Memoria Flash Player Con Valores

El segundo grupo son los objetos, que transmiten referencias. Si ejecutamos el siguiente código:

```
Var a:Object;
Var b:Object;
a = new Object();
a.variable = 5;
b = a;
b.variable = 7;
```

Se crearía una única instancia Object con dos referencias o formas para el objeto. Desde un nivel muy alto, la memoria de Flash Player sería como aparece en la figura 7.3:

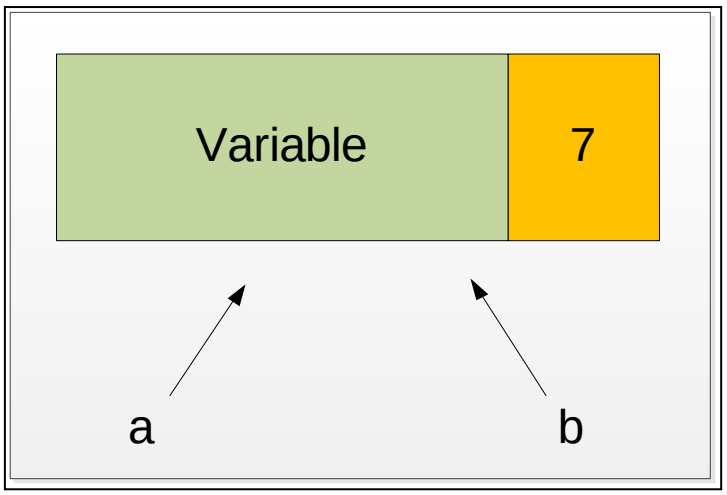


Figura 7.3 Funcionamiento De La Memoria Flash Player Con Referencias

Primero ponemos dos variables con los nombres a y b. Ambas variables son de tipo Object. Primero creamos una nueva instancia Object y la asignamos a la variable a. Ahora puede que a se refiere al nuevo objeto. Cuando configuramos a.someVar al valor 5 estamos configurando esa propiedad dentro del objeto al que se refiere a. Después asignamos b. Esto no hace una copia del objeto, sino que solamente nos dice que a y b se refieren al mismo objeto.

Para terminar, cuando configuramos b.someVar al valor 7, estamos configurando la propiedad dentro del objeto al que se refieren tanto a como b. Ya que a y b se igualan al mismo objeto, a.someVar es igual que b.someVar.

7.2.1.3. Recolección de elementos no utilizados en Flash Player

La recolección de elementos no utilizados es un procedimiento que pide memoria que ya no está en uso para que esta pueda volver a ser utilizada por otra aplicación o, en ciertos casos, se devuelve al sistema operativo. La recolección de elementos no útil se produce de forma automática durante la asignación. Esto

quiere decir que la recolección de elementos utilizados no se produce cuando la memoria no está en uso, sino que más bien se produce cuando la aplicación pide más memoria. Aquí es cuando el proceso responde a la recolección de elementos no usados, llamado Garbage Collector, e intenta reclamar memoria disponible para la reasignación.

El Garbage Collector sigue un proceso de dos partes para determinar qué parte de la memoria ya no está en uso. Al entender este procedimiento nos da la visión necesaria para poder desarrollar aplicaciones que utilicen la memoria correctamente y para la información presentada por el Profiler de Flex.

La primera parte del procedimiento de la recolección de elementos no utilizados se denomina *reference counting* (recuento de referencias) y la segunda se denomina *mark and sweep* (marcar y barrer). Ambas dependen de distintos métodos para garantizar que la memoria en cuestión ya no sea referenciada por otros objetos en uso.

Como hemos visto, cuando creamos un nuevo objeto normalmente se crea también una referencia o una forma de referirse a ese objeto. Si observamos esta pequeña parte de código, podemos ver que creamos una referencia llamada canvas a nuestra nueva instancia Canvas y una con el nombre Lbl al nuevo label creado. También añadimos label como un nodo descendiente de Canvas.

```
Var canvas:Canvas = new canvas ();

Var LblLabel = new label ();

canvas.addChild(Lbl);
```

Todos los componentes conservan una referencia a sus nodos descendientes y todos los componentes descendientes mantienen una referencia a su nodo ascendiente. Esto significa que las referencias del fragmento de código anterior sería como la figura 7.4:

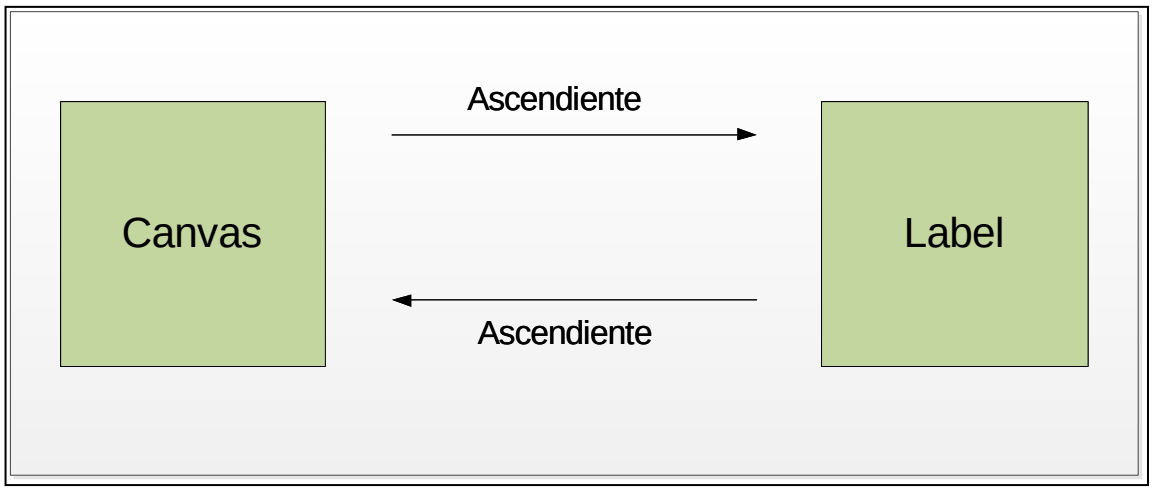


Figura 7.4 Relación De Las Referencias De Los Componentes Canvas y Label

El fragmento de código anterior demuestra por lo menos cuatro referencias que tenemos que tener en cuenta.

- canvas es una referencia a una instancia de Canvas.
- lbl es una referencia a una instancia de Label.
- lbl.parent es una referencia a la instancia Canvas.
- canvas.getChildAt (0) devuelve una referencia a la instancia Label.

La siguiente aplicación se utiliza para ilustrar ambas partes del procedimiento de recolección de elementos no utilizados y

determina los elementos que se encuentran libres para la recolección.

```
<?xmlversión="1.0" enconding="utf-8"?>
<mx:ApplicationxmlIns:mx=http://www.adobe.com/2006/mxml
creationComplete="onCreation(event)">
<mx:Script>
<![CDATA[
import                mx.controls.TextInput;
import                mx.controls.Label;
import                mx.controls.ComboBox;
import                mx.containers.Canvas;
import                mx.controls.DataGrid;
import                mx.containers.Vbox;
import                mx.containers.Hbox;

private function onCreation(event:Event):void {
var hBox:HBox = new HBox();

var vBox:VBox = new VBox();
vBox.addChild (new DataGrid ());
vBox.addChild (new ComboBox());
hBox.addChild (vBox );
```

```
this.addChild (hBox );

var canvas:Canvas = new Canvas ();

canvas.addChild (new Label () );

var textInput:TextInput = new TextInput ();

}

]]>

</mx:Script>

</mx:Application>;
```

La aplicación llama al método onCreation() en el instante en que se produce el evento creationComplete. Este método crea un nuevo Hbox con un Vbox dentro de él. El Vbox contiene un DataGrid y una instancia ComboBox. El Hbox se añade más tarde como un nodo descendiente de la aplicación. Entonces se creará una instancia Canvas con una instancia Label como nodo descendiente y por último se crea una instancia de TextInput.

Sin embargo, es importante tener en cuenta que ni la instancia Canvas ni TextInput se añaden nunca a la aplicación mediante el método addChild(). En la figura 7.5 aparecen las referencias que son importantes justo antes de salir del método onCreation().

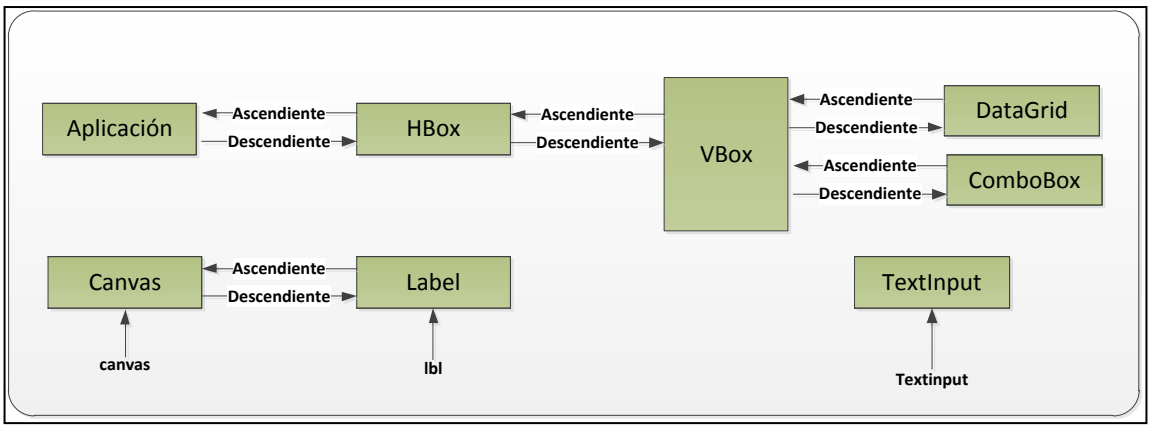


Figura 7.5 Referencias Del Método onCreation()

En el diagrama aparecen las referencias creadas en el método onCreation(), incluyendo las variables canvas, lbl, hBox y textInput. Sin embargo, estas variables se definen dentro del método onCreation(). Esto nos indica que las variables son locales al método; una vez que el método termina su ejecución, esas referencias desaparecerán pero los objetos creados dentro de este método continuarán existiendo.

Como hemos visto anteriormente, las referencias son los números que utiliza *Garbage Collector* para determinar las partes de la memoria que pueden ser reclamadas. Si no tenemos una referencia a un objeto, no hay forma de acceder a sus propiedades. Si no hay forma de acceder al objeto, entonces Garbage Collector reclama la memoria que utiliza para ocuparla.

El primer método que utiliza el *Garbage Collector* para determinar si existen referencias a un objeto se denominan reference counting (recuento de referencias). Es el método más sencillo y rápido para determinar si un objeto puede ser recolectado. Cada vez que se crea una referencia a un objeto, Flash Player incrementa un

contador asociado con el objeto. Cuando se elimina una referencia, el contador disminuye. Si el contador es cero, el objeto es un candidato para la recolección de elementos no utilizados.

Si miramos el ejemplo de código veremos el único objeto con un contador de referencia cero es TextInput. Una vez que el método onCreation() está completo, TextInput se queda sin referencias. Entonces nuevamente, si no hay forma de hacer referencia a un objeto, entonces el objeto puede ser recolectado. Este ejemplo también revela un problema con el contador de referencias: la referencia circular.

Si examinamos Canvas y Label cada una tiene una referencia hacia el otro, lo cual significa que cada uno tiene un contador de referencia mayor que cero. Sin embargo, no hay ninguna referencia a estos componentes en la aplicación. Si Flash Player pudiera identificar este hecho, la memoria para ambos componentes podría ser reclamada. Ésta es la razón para el segundo método utilizado dentro del procedimiento para recolectar elementos no utilizados, *mark* and *sweep* (marcar y barrer).

Utilizando este método, Flash Player empieza en el nivel superior de la aplicación y marca cada objeto con el que se encuentra una referencia. Después se desplaza hacia abajo en cada objeto y repite el proceso, continuando su inmersión hasta que termina con los objetos que tiene que marcar. Al final de este proceso, ni Canvas ni Label estarán marcados y serían candidatos para la recolección de elementos no utilizados. Aunque este método produce resultados definitivos, son muy bajos comparados con reference counting ya que no se ejecuta continuamente. Trabajando juntos, estos dos métodos pueden utilizarse para alcanzar niveles superiores de funcionamiento y precisión de recolección de elementos no utilizados.

7.2.1.4. Recolección de elementos no utilizados

Ahora que entendemos cómo decide *Garbage Collector* a la hora de reclamar la memoria, podemos empezar a usar *Garbage Collector* y dejar que este haga su trabajo. Esto quiere decir que se tiene que comprobar que se elimina todas las referencias a un objeto cuando ya no sea necesario. Dejar una referencia accidental a un objeto evita que la memoria sea reclamada; la imposibilidad para reclamar esta memoria puede ocasionar que la memoria continúe creciendo a medida que la aplicación continúa ejecutándose. A esto se denomina comúnmente *memory leak* (fuga de memoria).

Como hemos visto, los descendientes visuales se añaden a los componentes utilizando el método addChild(). Los descendientes también pueden eliminarse utilizando el método removeChild() o remoteChildAt(). El primer método necesita que le proporcionemos nuestra propia referencia al descendiente que se va a eliminar, como removeChild(hBox), mientras que el segundo método le permite especificar el índice del descendiente dentro de su ascendiente, removeChildAt(0). El segundo sencillamente utiliza la referencia mantenida por el ascendiente para identificar el descendiente. Estos métodos nos aseguran que tanto las referencias del ascendiente como del descendiente se eliminan desde dentro de los componentes.

En el ejemplo anterior, si eliminábamos el HBox desde la aplicación con el método removeChild(), el HBox y todos los objetos contenidos dentro de él sencillamente estarían disponibles para la recolección, ya que no hay otras referencias al descendiente. Hay otras formas de crear y mantener las referencias además de las variables y las propiedades que hemos tratado hasta ahora.

De hecho, la causa más frecuente de las fugas de memoria a la hora de programar en Flex es la utilización de escuchadores de eventos sin el cuidado necesario.

7.2.1.5. Entender las fugas ocasionadas por los escuchadores de eventos

Trataremos ahora sobre el método addEventListener() que nos permite escuchar de forma programática un evento transmitido.

Los objetos que tienen que notificarse cuando se produce un evento se registran ellos mismo como escuchadores. Lo hacen llamando al método addEventListener() en el objeto que transmite el evento (llamado difusor o distribuidor). En el siguiente ejemplo aparece un caso sencillo:

```
Var textinput

Textinput = new textinput();

textinputaddEventListener(`change´, handleTextChanged);
```

En este caso se espera que Textinput transmita un evento con el nombre change, en algún punto en el futuro y queremos que el método handleTextChanged() en la instancia TextInput,

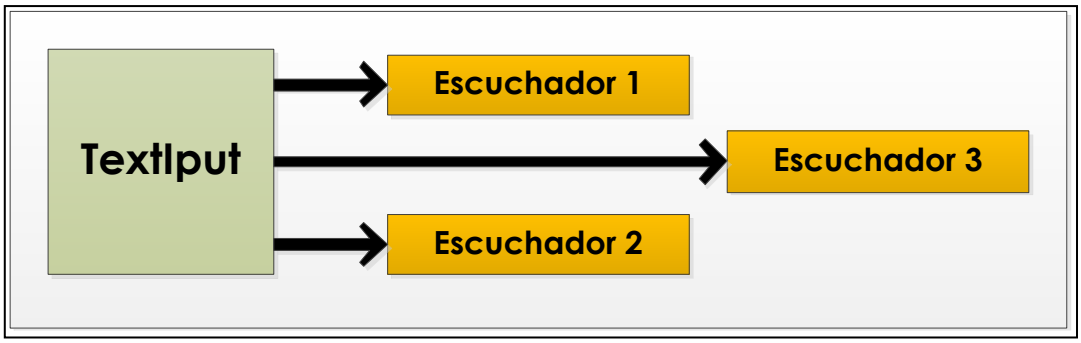


Figura 7.6 Instancia TextInput notifica a cada objeto que se registró como escuchador

responde añadiendo una referencia al objeto (la que contiene el método handleTextChanged) en una lista de objetos que tienen que ser notificados cuando este evento se produzca. Cuando llega la hora de transmitir el evento change, la instancia TextInput itera a través de esta lista y notifica a cada objeto que se ha registrado como escuchador. En memoria sería algo como lo que aparece en la siguiente figura 7.6:

El aspecto importante con el que hay que quedarse es que cada objeto que transmite un evento mantiene una referencia a cada objeto que escucha el evento que se va a transmitir. En términos de recolección de elementos no utilizados esto significa que, en determinadas circunstancias, si un objeto está escuchando eventos, nunca estará disponible para la recolección de elementos no utilizados.

El arma principal para combatir este problema es la diligencia. De igual modo que cualquier descendiente se añade con addChild() puede eliminarse con removeChild(), addEventListner() tiene una función paralela llamada

removeEventListener() que detiene que un evento sea escuchado.

Cuando se llama a removeEventListener() también se elimina la referencia al escuchador que mantiene el difusor, con lo cual se libera de forma potencial el escuchador para la recolección de elementos no utilizados.

En un mundo ideal, el número de llamadas realizadas a addEventListener() y removeEventListener() en una aplicación deberían ser iguales. Sin embargo, hay ocasiones en las que tenemos menos control sobre los objetos cuando ya no son necesarios y usar removeEventListener() sencillamente no es factible. En estas situaciones, podemos utilizar un concepto llamado weak references(referencias débiles).

7.2.1.6 Utilizar las referencias débiles con escuchadores

Cuando se añade un escuchador de eventos a un difusor, el desarrollador pude especificar que el escuchador de eventos tendría que utilizar referencias débiles.

Esto se consigue especificando parámetros adicionales para el método addEventListener().

```
var textInput:TextInput = new TextInput ();

textInput.addEventListener ('change',handleTextChanged,
false, 0 , true) ;
```

Anteriormente, indicábamos cómo utilizar los primeros dos parámetros del método addEventListener(): el nombre del evento y el método para llamar cuando el evento se produzca. Sin embargo, hay otros tres parámetros que pueden especificarse. En

orden, estos parámetros especifican si el escuchador de eventos utilizará la captura, su prioridad relativa a otros escuchadores para este evento y finalmente si tendrían que utilizarse referencias débiles.

Especificar un valor true (por defecto es false) para el quinto parámetro del método addEventListener() especifica que la referencia establecida por el escuchador se considera débil. Todas las referencias que hemos tratado hasta el momento en este capítulo se consideran referencias fuertes, que diremos que son referencias que Garbage Collector tiene en cuenta a la hora de decidir si un objeto está disponible para la recolección. Al contrario, las referencias débiles son ignoradas por Garbage Collector, lo cual significa que un objeto sólo con referencias débiles será recolectado.

Como un ejemplo más concreto, si un objeto tiene una referencia fuerte y tres referencias débiles, no será recolectado como elemento no utilizado. Sin embargo, si se ha eliminado la referencia fuerte, las referencias débiles serán ignoradas y el objeto podrá ser recolectado. En la práctica, esto significa que especificar la señal de referencia débil en las llamadas addEventListener() casi siempre es una buena práctica. Evita el caso en el que escuchar un evento sea la única razón de que un objeto no sea recolectado.

7.2.3. Análisis de la memoria en una aplicación Flex

El análisis de la memoria implica examinar la memoria utilizada, además de la memoria que está actualmente en uso por los objetos de una aplicación. Esos objetos podrían ser sencillamente clases, como Strings, u objetos visuales complejos, como DataGrid; Utilizando el análisis de memoria se puede determinar si existe un número apropiado de objetos y si esos objetos utilizan una cantidad apropiada de memoria.

7.2.3.1. Análisis de la aplicación QuickDelivery

En el siguiente ejemplo utilizaremos las posibilidades de análisis de memoria de Flex Profiler para identificar una fuga de memoria en la aplicación Profiler Test. Es una herramienta incluida en el Flex Builder que nos permite realizar sesiones de profiling (para optimizar el rendimiento y el uso de recursos)

Capturaremos y revisaremos instantáneas de memoria, identificaremos las relaciones de referencia entre objetos y determinaremos el objeto responsable de la fuga de memoria.

1. Hacemos clic en el botón **Profile Application** (Analizar aplicación) que está inmediatamente a la derecha el botón **Debug** (Depurar, Figura 7.7).

Figura 7. 7 Botón Profile Application (Analizar aplicación)

   La aplicación empieza a ejecutarse, pero el foco volverá a la ventana de Flex Builder y aparecerá una ventana *Configure Profiler*.

2. Comprobamos que las opciones Enable memory profiling (Activar análisis de memoria), Watch live memory data (Buscar datos de memoria viva) y Genérate object allocation stack traces (Generar rastros de pila en la ubicación del objeto) están seleccionadas.

   El análisis de la memoria reducirá de forma drástica la velocidad de la aplicación a medida que recolecta cantidades

significativas de datos sobre cuándo, dónde y cómo se crean los objetos.

Estas opciones sólo tienen que seleccionarse cuando se intente diagnosticar una fuga de memoria o verificar que una fuga no existe (figura 7.8).

3. Hacemos clic en Resume (Continuar). La perspectiva de Eclipse cambia a Flex Profiling.

4. La perspectiva *Profiling* empieza a visualizar la información sin tener en cuenta la utilización de memoria, las instancias de objetos acumulados *y* actuales *y la* memoria acumulada y actual en utilización para cada tipo de objeto (figura 7.9).

   La esquina superior izquierda de la pantalla muestra la aplicación que está actualmente siendo analizada.

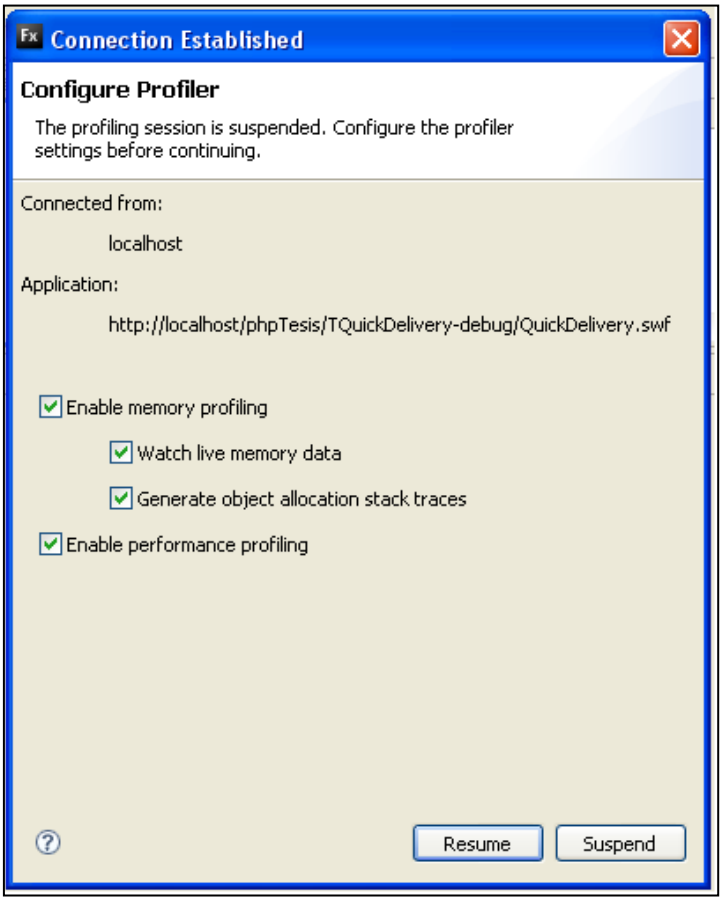
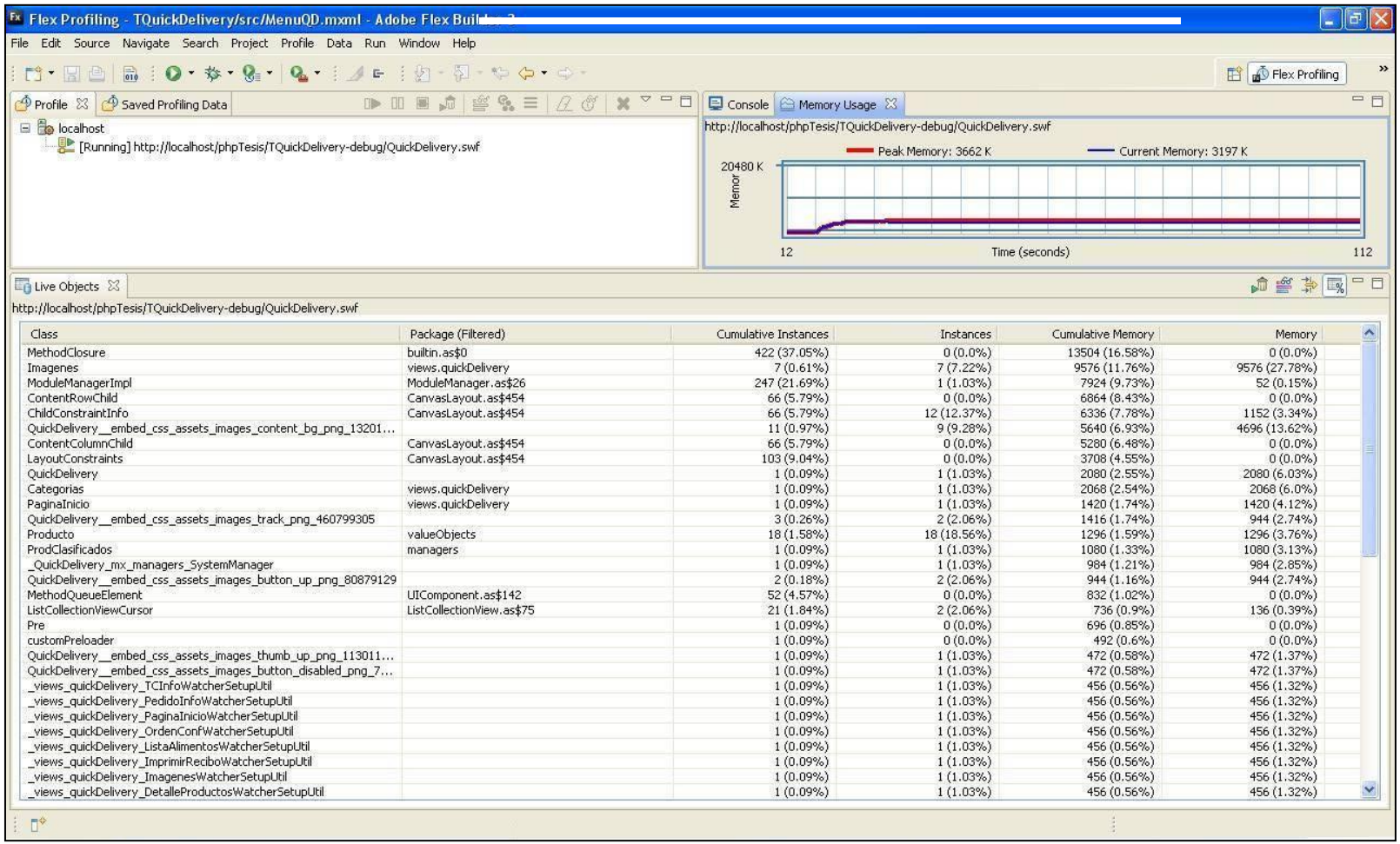


Figura 7. 8 Configuración De Las opciones De profiler



Class	Package (Filtered)	Cumulative Instances	Instances	Cumulative Memory	Memory
MethodClosure	builtin.as$0	422 (37.05%)	0 (0.0%)	13504 (16.58%)	0 (0.0%)
Imagenes	views.quickDelivery	7 (0.61%)	7 (7.22%)	9576 (11.76%)	9576 (27.78%)
ModuleManagerImpl	ModuleManager.as$26	247 (21.69%)	1 (1.03%)	7924 (9.73%)	52 (0.15%)
ContentRowChild	CanvasLayout.as$454	66 (5.79%)	0 (0.0%)	6864 (8.43%)	0 (0.0%)
ChildConstraintInfo	CanvasLayout.as$454	66 (5.79%)	12 (12.37%)	6336 (7.78%)	1152 (3.34%)
QuickDelivery__embed_css_assets_images_content_bg_png_13201...		11 (0.97%)	9 (9.28%)	5640 (6.93%)	4696 (13.62%)
ContentColumnChild	CanvasLayout.as$454	66 (5.79%)	0 (0.0%)	5280 (6.48%)	0 (0.0%)
LayoutConstraints	CanvasLayout.as$454	103 (9.04%)	0 (0.0%)	3708 (4.55%)	0 (0.0%)
QuickDelivery		1 (0.09%)	1 (1.03%)	2080 (2.55%)	2080 (6.03%)
Categorias	views.quickDelivery	1 (0.09%)	1 (1.03%)	2068 (2.54%)	2068 (6.0%)
PaginaInicio	views.quickDelivery	1 (0.09%)	1 (1.03%)	1420 (1.74%)	1420 (4.12%)
QuickDelivery__embed_css_assets_images_track_png_460799305		3 (0.26%)	2 (2.06%)	1416 (1.74%)	944 (2.74%)
Producto	valueObjects	18 (1.58%)	18 (18.56%)	1296 (1.59%)	1296 (3.76%)
ProdClasificados	managers	1 (0.09%)	1 (1.03%)	1080 (1.33%)	1080 (3.13%)
_QuickDelivery_mx_managers_SystemManager		1 (0.09%)	1 (1.03%)	984 (1.21%)	984 (2.85%)
QuickDelivery__embed_css_assets_images_button_up_png_80879129		2 (0.18%)	2 (2.06%)	944 (1.16%)	944 (2.74%)
MethodQueueElement	UIComponent.as$142	52 (4.57%)	0 (0.0%)	832 (1.02%)	0 (0.0%)
ListCollectionViewCursor	ListCollectionView.as$75	21 (1.84%)	2 (2.06%)	736 (0.9%)	136 (0.39%)
Pre		1 (0.09%)	0 (0.0%)	696 (0.85%)	0 (0.0%)
customPreloader		1 (0.09%)	0 (0.0%)	492 (0.6%)	0 (0.0%)
QuickDelivery__embed_css_assets_images_thumb_up_png_113011...		1 (0.09%)	1 (1.03%)	472 (0.58%)	472 (1.37%)
QuickDelivery__embed_css_assets_images_button_disabled_png_7...		1 (0.09%)	1 (1.03%)	472 (0.58%)	472 (1.37%)
_views_quickDelivery_TCInfoWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_PedidoInfoWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_PaginaInicioWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_OrdenConfWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_ListaAlimentosWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_ImprimirReciboWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_ImagenesWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)
_views_quickDelivery_DetalleProductosWatcherSetupUtil		1 (0.09%)	1 (1.03%)	456 (0.56%)	456 (1.32%)

Figura 7. 9 Perspectiva Flex Profiling

La esquina superior derecha muestra un gráfico de la utilización de memoria actual junto con el pico de memoria utilizada. Un indicador claro de una fuga de memoria en una aplicación es que la memoria nunca tiene picos. Si la aplicación continúa creciendo con su uso continuado en el tiempo, seguramente tendrá fugas de memoria.

La parte inferior de la pantalla contiene en la actualidad una vista llamada Live Objects. Esta vista muestra la clase, paquete e instancias acumuladas y actuales de cada tipo de objeto, además de la memoria acumulada por estos objetos.

En el lado derecho de la pantalla al mismo nivel que la pestaña Live Objects se verá una serie de íconos.

5. Cambiamos al navegador que está ejecutando la aplicación Quick Delivery y hacemos clic en la categoría Lácteos.
   Con el analizador de memoria activado, toda la aplicación se mueve de forma mucho más lenta que antes.

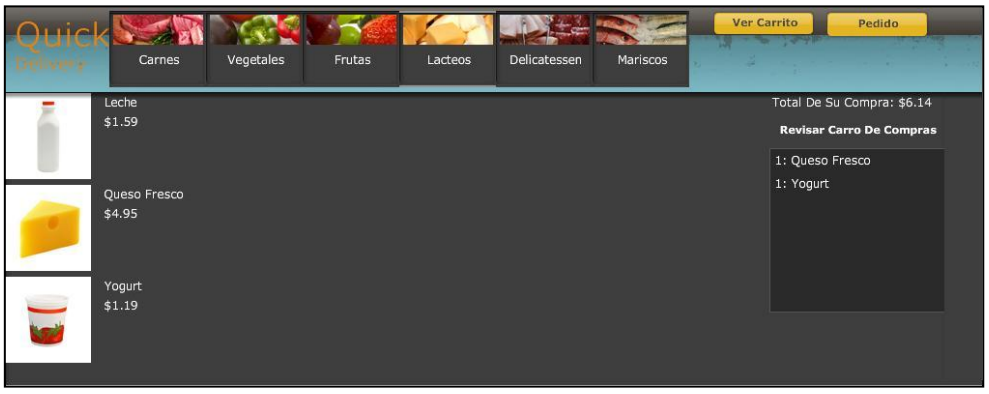


Figura 7. 10 Aplicación QDelivery, Categoría Lácteos

6. Una vez que los productos de la categoría Lácteos aparecen, cambiamos a Flex Builder y observaremos la vista Live Objects. En la columna que está más a la izquierda veremos las instancias acumuladas.

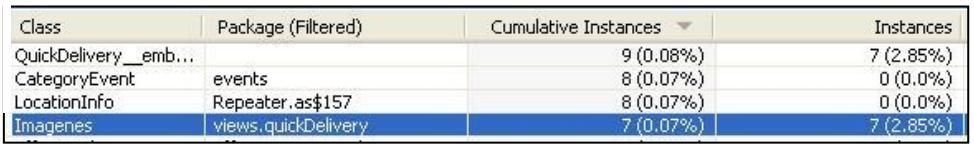
Ahora se ve la clase Imágenes que aparece en la lista que hay en la columna de la izquierda. Las instancias acumuladas y las instancias deberían indicar 7.

Cuando se hace clic en la categoría Lácteos, la aplicación crea un nuevo HTTPService y carga datos XML para esa categoría. Los datos se transmiten entonces a la propiedad dataProvider del repetidor, que crea una instancia de Imágenes para cada producto en la categoría.

La columna **Cumulative Instances** muestra el número total de veces que se han creado instancias de esta clase desde que se inició la aplicación. La columna **Instances** muestra el número de instancias que todavía están en la memoria en ese momento. La diferencia entre la columna Cumulative Instances e Instances es el número de instancias que han sido recolectadas como elementos no utilizados en algún punto.

7. Hacemos clic en cada una de las categorías de la aplicación para visualizar los productos contenidos dentro de ellas. Cambiamos a Flex Profiler y visualizamos la información de las instancias para la clase Imágenes.

   La clase Imágenes mostrará 7 instancias acumuladas y actuales. Esto indica estas instancias han sido recolectadas como elementos no utilizado todavía. Esto nos muestra claramente que en nuestra aplicación no existe una fuga de memoria en la clase Imágenes.



Class	Package (Filtered)	Cumulative Instances	Instances
QuickDelivery__emb...		9 (0.08%)	7 (2.85%)
CategoryEvent	events	8 (0.07%)	0 (0.0%)
LocationInfo	Repeater.as$157	8 (0.07%)	0 (0.0%)
Imagenes	views.quickDelivery	7 (0.07%)	7 (2.85%)

Figura 7. 11 Clase Imágenes

8. Hacemos clic en cada una de las categorías de la aplicación para visualizar los productos contenidos dentro de ellas.

Cambiamos a Flex Profiler y visualizamos la información de las instancias para la clase Producto.

La clase Producto mostrará 18 instancias acumuladas y actuales.

Class	Package (Filtered)	Cumulative Instances	Instances
QuickDelivery__emb...		3 (0.16%)	2 (1.61%)
Producto	valueObjects	18 (0.99%)	18 (14.52%)
ListCollectionViewCu...	ListCollectionView.as$75	37 (2.03%)	5 (4.03%)

Figura 7. 12 Clase Producto Con 18 Instancias

Cuando navegamos a través de las categorías y productos, vemos que la clase Producto nos muestra 22 instancias acumuladas y actuales, esto podía indicar una fuga de memoria.

Class	Package (Filtered)	Cumulative Instances	Instances
QuickDelivery__emb...		27 (0.16%)	0 (0.0%)
Producto	valueObjects	22 (0.13%)	22 (15.6%)
CursorQueueItem	CursorManagerImpl.as$682	12 (0.07%)	0 (0.0%)

Figura 7. 13 Clase Producto Con 22 Instancias

9. Encima de la columna llamada Memory en la cuadrícula Live Objects, hay una serie de iconos organizados de forma horizontal. El primero de estos iconos hace que Garbage Collector se ejecute inmediatamente. Hacemos clic en este icono ahora (figura 7.14).

Figura 7. 14 Run Garbage Collector

Como se ha mencionado anteriormente en esta sección la recolección de elementos no utilizados se ejecuta de forma automática durante la asignación; sin embargo, es difícil para el desarrollador entender de forma precisa cuándo se ha ejecutado por última vez la recolección de elementos no

utilizados. Flex Profiler le proporciona la posibilidad de ejecutar el recolector de elementos no utilizados cuando sea necesario.

10. Revisamos ahora las instancias acumuladas y las actuales para la clase Producto de nuevo. El número de instancias es ahora de 22. Como se ha ejecutado la recolección de elementos no utilizados de forma contundente y solamente hay 19 instancias de Producto en la pantalla actual, ahora parece cierto que hay un problema. Las instancias de Producto que ya no están en uso no son recolectadas. Como sabemos, lo único que evitaría la recolección de elementos no utilizados sería una referencia a estas instancias de Producto que aún permanece.

11. Hacemos clic en el ícono que hay justo a la derecha del ícono de recolección de elementos no usados para tomar una instantánea de la memoria (Figura 7.15).

Figura 7. 15 Take Memory Snapshot

Una instantánea de memoria guarda el estado actual de la memoria en el momento en que se hace clic en el botón Take Memory Snapshot(Tomar instantánea de la memoria). La instantánea no actualiza la vista Live Object, pero le permite analizar la memoria de una forma mucho más profunda.

12. Hacemos doble clic en las palabras Memory Snapshot que hay debajo de la aplicación que se está ejecutando en la esquina superior izquierda de la ventana. Tras analizar los datos, se abre una nueva pestaña después de la pestaña Live Object con la información de la instantánea. Es posible tener múltiples instantáneas e incluso guardar estos datos para revisarlos cuando intente resolver problemas dentro de la aplicación. En la figura

7.16 aparecen los resultados de una nueva instantánea de memoria.

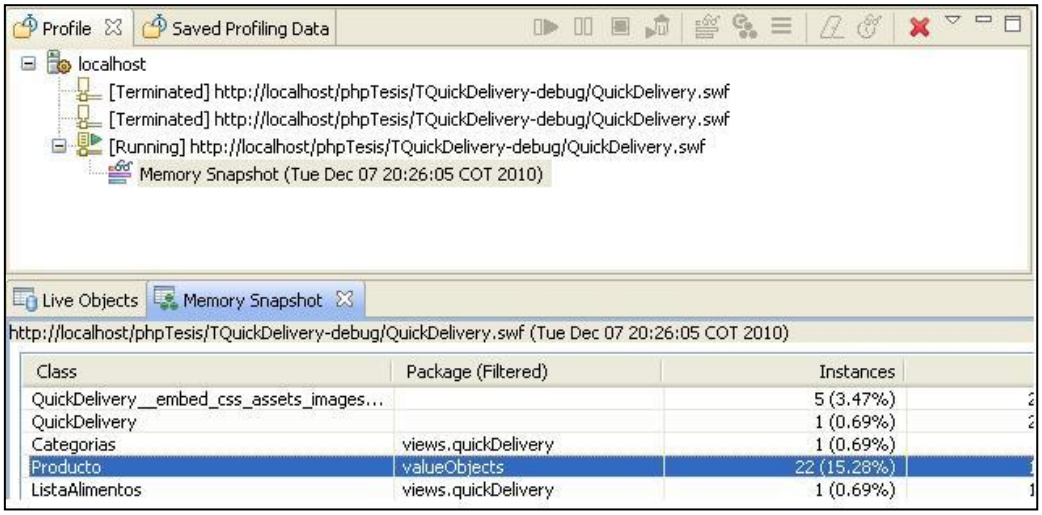


Figura 7. 16 Pestaña Memory Snapshot

Cuando se toma una instantánea de la memoria, Flex Profiler ejecuta de forma implícita la recolección de elementos no utilizados en primer lugar.

13. Dentro de la pestaña Memory Snapshot haremos doble clic a continuación en la clase Producto. Podemos ver 22 listas distintas para cada objeto Producto en la memoria cuando se tomó esta instantánea, cada una con un número entre paréntesis. El número que aparece entre paréntesis es el número total de referencias al objeto actual.

    Haremos clic en la primera etiqueta valueObjects:Producto.
    El lado derecho de la pantalla muestra Allocation trace. Esta información muestra dónde se ha creado por primera vez este objeto en el flujo del programa. En este caso veremos que este objeto en particular se creó en una aplicación llamada ProdClasificados.mxml. También se registra el número de línea de cada una de estas llamadas.

En este punto sabemos que el objeto fue creado debido a la llamada ProdClasificados.mxml, que es correcta, pero no sabemos qué otro objeto tiene una referencia que está evitado que este objeto sea recolectado.

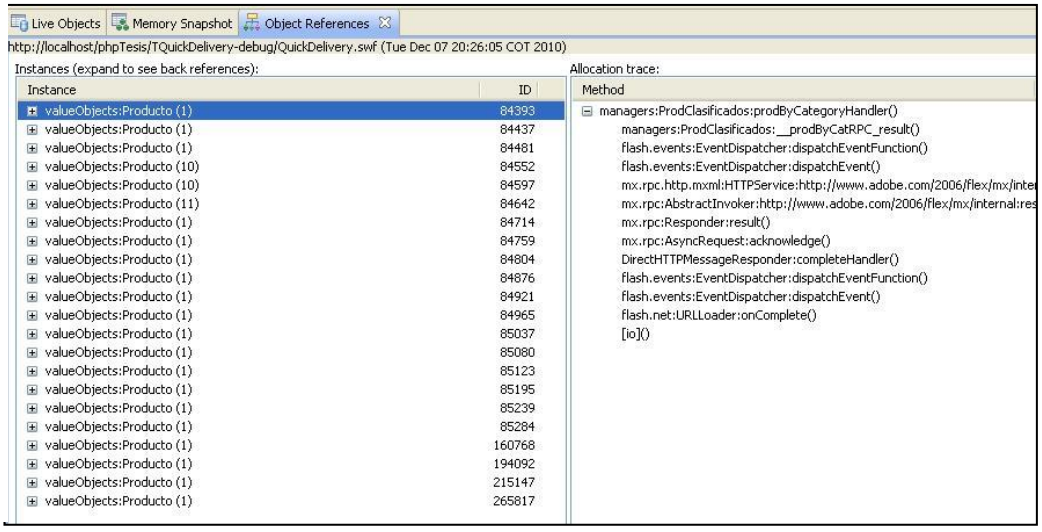


Figura 7. 17 Pestaña Object References

14. Hacemos clic en el indicador de árbol abierto que hay a continuación de la etiqueta valueObjects:Producto.

    Esto hace que el árbol se abra y muestre todos los objetos del sistema que contienen las referencias al objeto actual y el nombre de la propiedad dentro de ese objeto que contiene la referencia. Esta lista puede ser extensiva. Como hemos visto, hay muchas referencias circulares entre los ascendientes y descendientes que no son factores determinantes durante la fase *mark and sweep* de la recolección de elementos no utilizados; por eso estamos buscando elementos aquí que todavía podrían ser un factor determinante.

    La fuga de memoria que estamos viendo implica objetos escritos en el código, ignoraremos cualquier objeto que exista en la ruta mx.*, debido a que estos objetos son proporcionados por adobe y nos vamos a centrar en los objetos que nosotros hemos creado.

15. Nos vamos a centrar en las referencias de los elementos restantes. Abriremos cada uno de los sub elementos llamados Function y examinaremos los descendientes de estas referencias. La mayoría de los descendientes dirán managers:ProdClasificados. Estas referencias existen principalmente como resultado de la vinculación de datos; sin embargo, no son de particular interés ya que sabemos que Garbage Collector puede manejar referencias circulares. También debería encontrar rápidamente una lista para managers:ProdClasificados y una propiedad de [listener()].

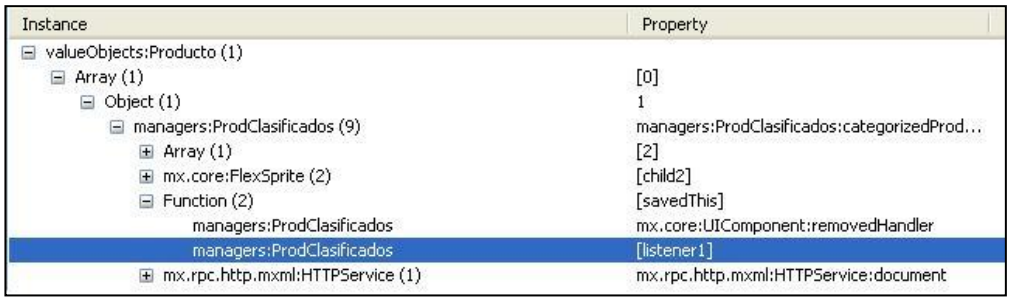


Figura 7. 18 Propiedad Listener() con su lista managers:ProdClasificados

16. Examinamos varias instancias de la clase Producto.
    Cada una de ellas nos mostrará un resultado parecido con una referencia desde managers:ProdClasificados [listener()].

17. Terminaremos esta sesión de análisis seleccionando la esquina superior izquierda de la pantalla y haciendo clic en el botón Stop o cerrando el navegador Web.

18. En este punto conocemos que managers:ProdClasificados [listener()] tiene una referencia a cada una de las clases Producto. El Profiler también le proporciona una pista de que esta referencia se debe a un escuchador de eventos.

Utilizando esta información, podemos ser capaces de encontrar rápidamente el problema.

7.2.3.2. Corregir la clase Producto

Arreglaremos la fuga de memoria identificada anteriormente modificando el escuchador de eventos para que utilice referencias débiles. Cambiamos a la perspectiva Flex Development (Desarrollo de Flex) en Flex Builder.

1. Abrimos la clase Producto.

2. Buscamos a continuación el código que añade un escuchador de eventos a la instancia managers:ProdClasificados.

3. En esta clase, comprobaremos que no hay llamada a removeEventListener(). Como se ha indicado antes, en teoría el número de llamadas addEventListener() debería corresponderse con el número de llamadas removeEventListener(). En este caso, hemos limitado el control a la clase ya no es necesaria, por lo que vamos a optar por hacer que la llamada addEventListener() utilice referencias débiles.

4. Cambiamos el código que añade el escuchador de eventos para que utilice referencias débiles:

   ProdClasificadosEventListener(´toggleViewOnly´,handleViewOnlyChanged, false, 0, true);

5. La referencia en la instancia managers:ProdClasificados creada para cada instancia de Producto no contará ya durante la recolección de elementos no utilizados.

6. Volvemos a Flex Builder y hacemos clic en el ícono Run Garbage Collector (Ejecutar recolector de elementos no utilizados).

7. La vista Live Objects debería mostrar ahora 22 instancias acumuladas pero sólo 18 instancias actuales, lo cual quiere decir que otras 4 han sido recolectadas y que esta fuga de memoria se ha arreglado.

7.2.4. Análisis del rendimiento de una aplicación Flex

El análisis del rendimiento se utiliza para buscar aspectos de la aplicación que no responden o donde puede mejorarse el rendimiento. Cuando se analiza el rendimiento generalmente se buscan métodos que se ejecutan de forma frecuente o métodos que necesitan largos períodos de tiempo cada vez que se ejecutan. La combinación de estos dos factores normalmente proporciona una buena indicación de dónde se debería utilizar el tiempo de optimización o de una potencial refactorización.

7.2.4.1. Analizar la aplicación QEstadisticas

1. Hacemos clic en el botón Profile Aplication.
2. Comprobaremos que solo la opción Enable perfomance profiling está seleccionada.

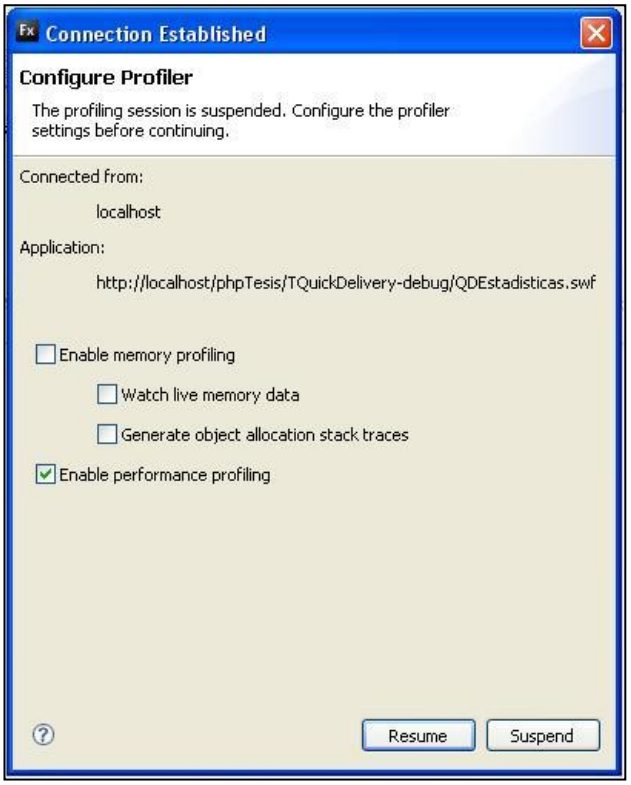


Figura 7.19 Configuración De Las opciones De Profiler

3. Hacemos clic en el botón Resume y la perspectiva Eclipse cambia a Flex Profiling.

4. Seleccionamos la aplicación que se está ejecutando y haga clic en el icono Reset Perfomance Data. Esto restablece las estadísticas de rendimiento.

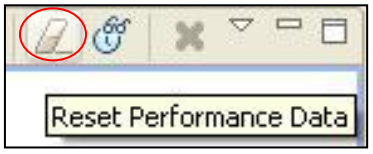


Figura 7.20 Reset Perfomance Data

En este caso, no estamos intentando evaluar el tiempo de inicio, por lo que estamos restableciendo las estadísticas de rendimiento después de que se haya iniciado pero antes de que interactuemos con ellas.

5. Cambiamos al navegador Web y navegaremos a través de cada una de las seis categorías y de las opciones de la aplicación.

   Comprobamos que espera a que se cargue los datos anteriores antes de cargar otros datos.

Figura 7. 21 Aplicación QDEstadística

6. Volvemos ahora a cambiar a Flex Profiler y a continuación seleccionamos la aplicación que se está ejecutando.

7. Hacemos clic ahora en el icono en forma de reloj y gafas para capturar un análisis de rendimiento (Capture Perfomance Profile).

   Aparece un análisis de rendimiento debajo de la aplicación que se está ejecutando, como ha hecho el análisis de memoria en la sección anterior.

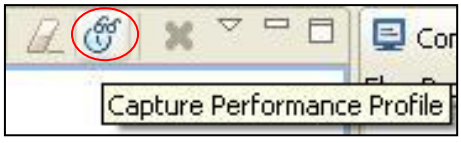


Figura 7.22 Capture Perfomance Profile

8. Hacemos clic en el icono en forma de cuadrado rojo para detener la aplicación que se ejecuta y después cerramos el navegador Web.

9. Hacemos después doble clic en el análisis de rendimiento para abrir la vista Perfomance Profile.

   La vista Performance Profile muestra el método y el paquete junto con otra información importante.

   - Calls: El número de llamadas a este método durante la captura.
   - Cumulative time: El tiempo acumulado dedicado a las llamadas a este método y cualquier método subsiguiente llamado dentro de este método.
   - Self Time: El tiempo acumulado dedicado a este método, pero no las subsiguientes llamadas al método dentro de este método.
   - Avg.Cumulative and Avg.Self: Los promedios de cada uno de los tipos de tiempo anteriores.

10. Hacemos clic en la columna Cumulative Time para clasificar del más largo al más corto (Figura 7.23).

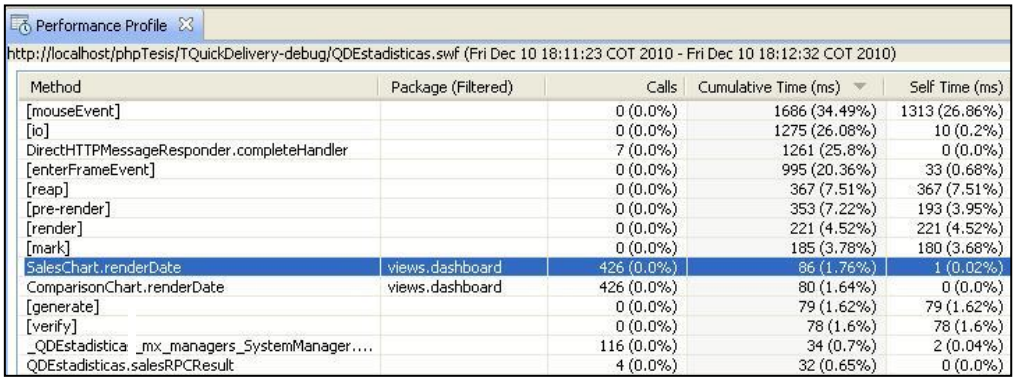


Figura 7. 23 Columna Cumulative Time (Tiempo acumulado)

Los tres elementos superiores son el completeHandler para HTTP Response, el método SalesChart.renderDate de QDEstadistica y el método ComparisonChart.renderDate. Sin embargo, el SelfTime para cada de estos métodos es casi inexistente. Los datos devueltos desde el servidor hacen que la aplicación cree una nueva instancia de la clase SalesChart.renderDate. Posiblemente estos métodos son los que están produciendo la pérdida de rendimiento.

7.2.5. Arreglar una clase con pérdida de rendimiento

Una vez que hemos detectado las causas de la pérdida de rendimiento por la creación de varias instancias, se puede utilizar una propiedad de la clase Repeater para limitar la recreación del objeto y aumentar de forma significativa la respuesta de la aplicación.

A esta clase llamamos a la propiedad recycleChildren la cual se configura a True. Cuando el repetidor está configurado a true, reutiliza los descendientes ya creados en vez de crear unos nuevos.

Veremos una reducción significativa en tiempo para estos métodos y un aumento del rendimiento.

Por último, el análisis del rendimiento trata sobre la optimización de los elementos que consumen la mayor cantidad de tiempo. Es un proceso interactivo que continúa hasta que la aplicación funciona como se espera.

La aplicación puede ser optimizada aún más repitiendo los pasos estudiados en estos ejercicios.

7.2.6. Pruebas en la aplicación web

7.2.6.1. Prueba de interfaz del usuario

Las actividades en esta prueba consisten en realizar las respectivas revisiones en cuanto a diseño y contenido visual, verificando que todo esté disponible al usuario sin ningún error.

1. Prueba De Carrito De Compras

A continuación se encuentran las pruebas realizadas al carrito de compras, verificando que este añadiendo correctamente los productos y el cálculo de la compra sea correcto.

Después de colocar los productos en el carro de compras nos dirigimos al mismo para verificar que los datos sean iguales al que se pueden observar en la figura 7.24.

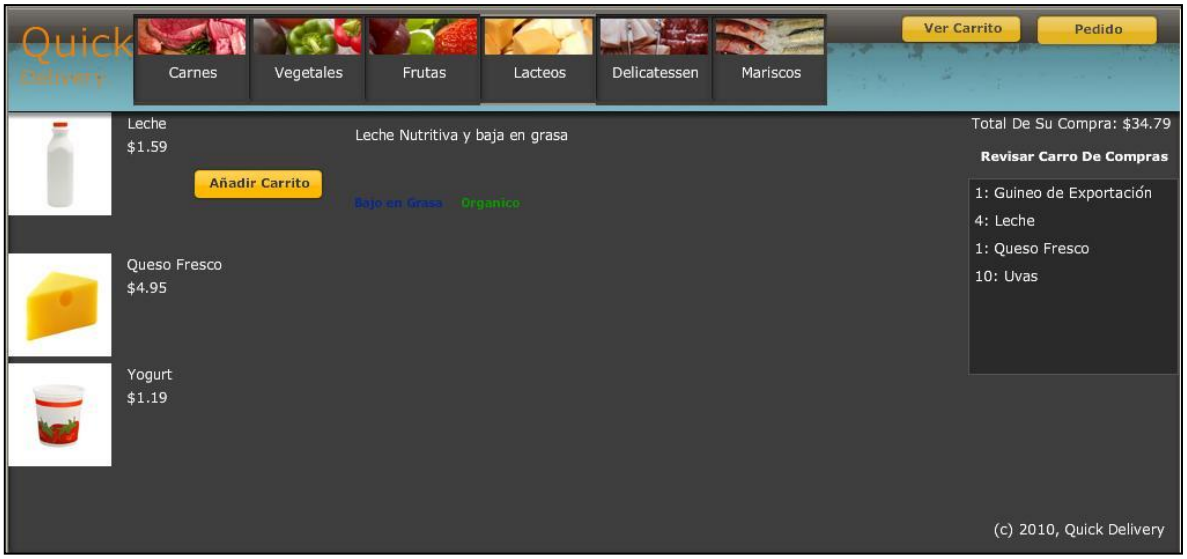
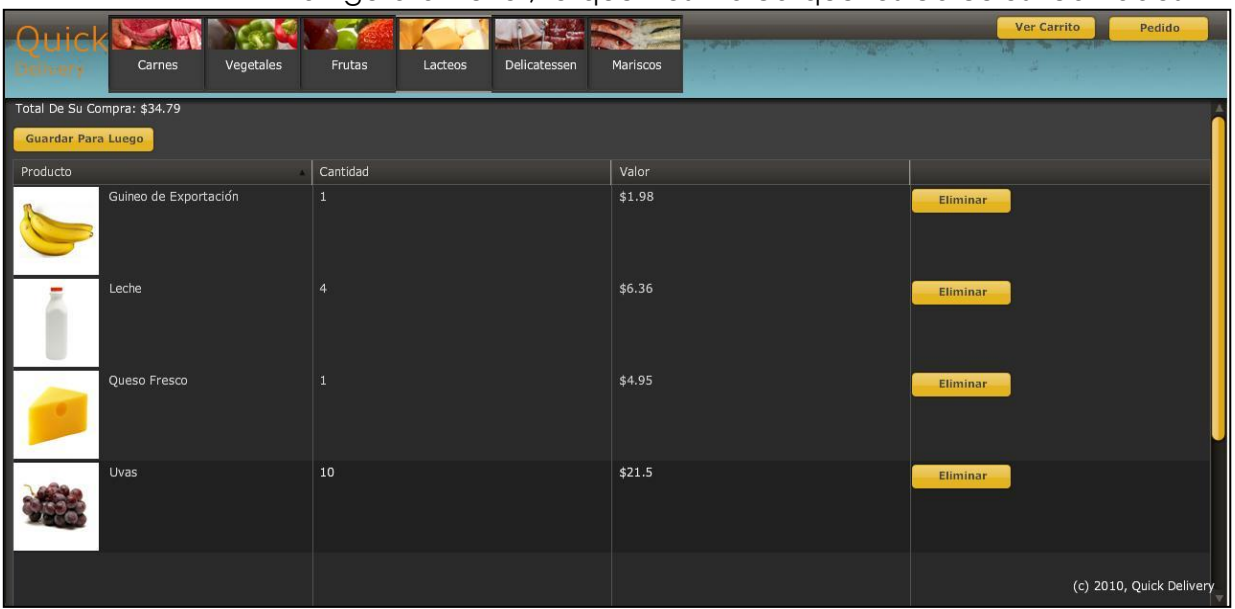


Figura 7. 24 Pantalla para añadir productos a carro de compras

Como observamos en la imagen 7.25 los valores son iguales a la figura anterior, lo que nos indica que los cálculos realizados



por el carro de compras son correctos. Además se realizó pruebas aumentando la cantidad, eliminando productos y utilizamos el botón "Guardar Para Luego", donde se verificó que al momento de guardar el carro de compras, cerrar la aplicación y luego al volverla a abrir encontramos los mismos datos, lo cual nos indica un correcto funcionamiento del carrito de compras.

Figura 7. 25 Pantalla de Carrito de compras

2. Prueba De Ingreso De Datos

Se constató que se las validaciones de los campos del formulario para el ingreso de datos del usuario estén validando correctamente los posibles errores que se dan al momento de ingresar la información. En caso de que algún campo no está llenado correctamente este se resaltará con recuadro rojo y verá aparecer un mensaje de error, para que así el usuario conozca cual es el error y lo pueda corregir. En la figura 7.26 podemos ver un error, este es debido a que no se llenó correctamente el campo de Email, y como vemos este esta resaltado y nos muestra un mensaje de error.

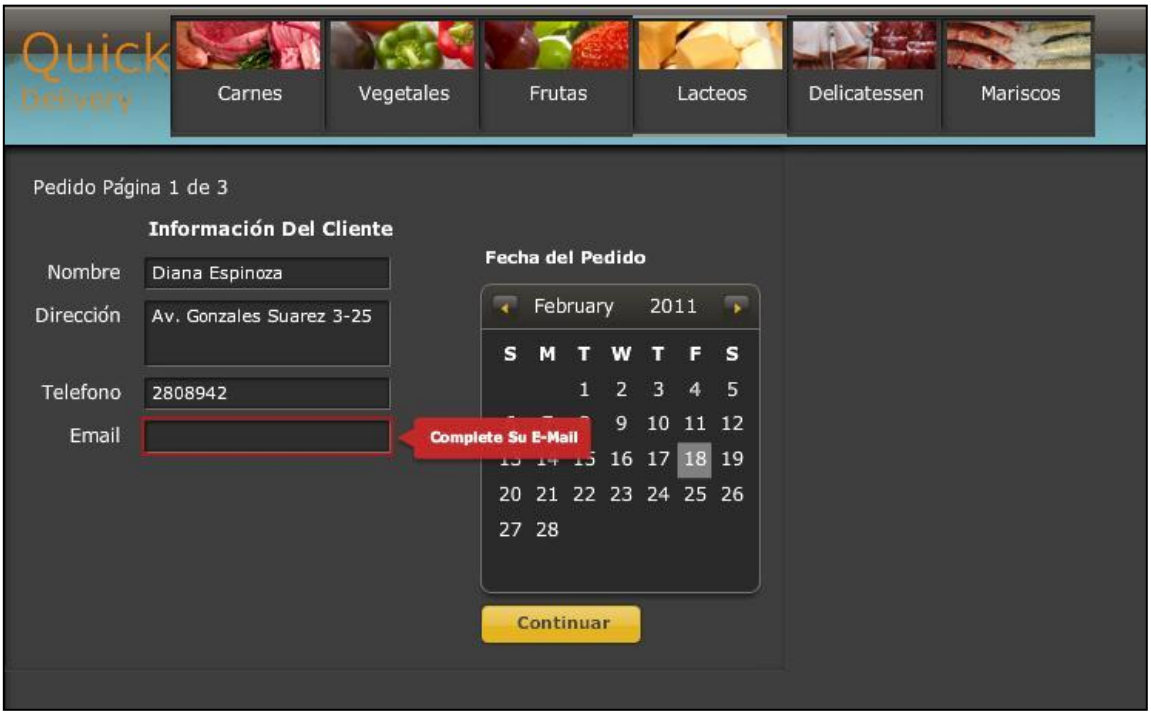


Figura 7. 26 Formulario De usuarios

7.2.6.2. Pruebas de compatibilidad

Las pruebas en cuanto a la compatibilidad están relacionadas con la ejecución de la aplicación web en varios entornos, en este caso en varios navegadores y sistemas operativos.

1. **Pruebas en navegadores**

   Se ejecutó la aplicación web en los navegadores Opera 11.1 y Safari 5.0.3, se obtuvo exitosos resultados en los dos navegadores realizando las mismas operaciones.

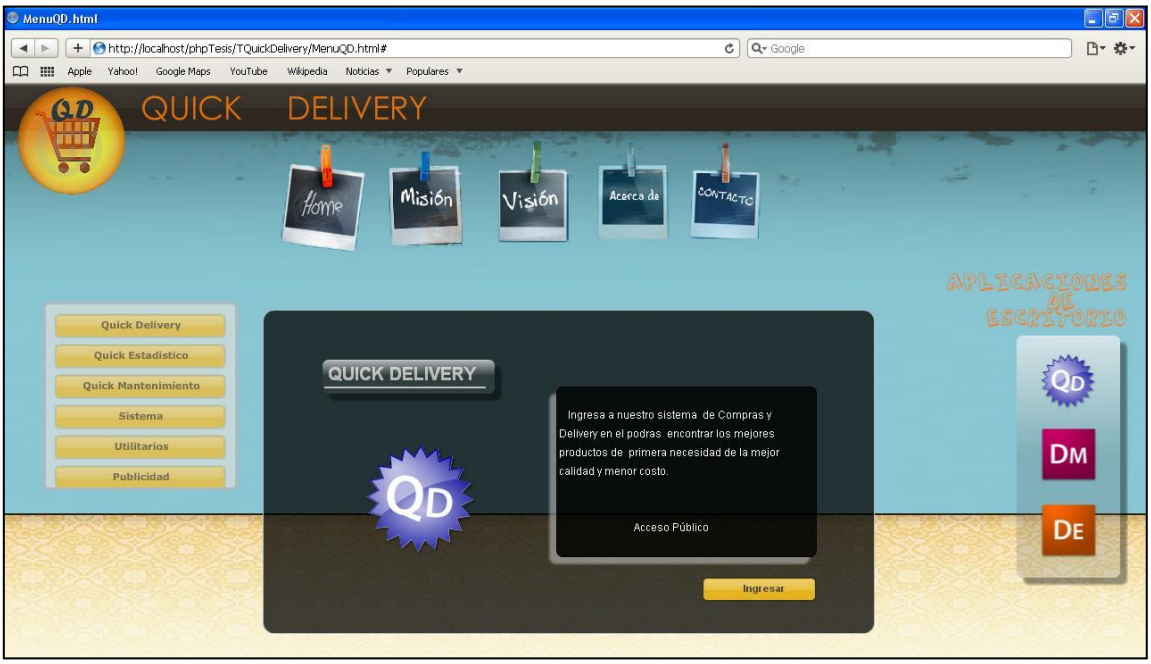


Figura 7. 28 Pantalla del sitio web ejecutada en Safari

2. Pruebas en sistema operativo

Anteriormente hemos visto trabajando a nuestra aplicación en el entorno de Windows, ahora mostraremos que la misma aplicación puede ser instalada y ejecutada en cualquier sistema operativo Mac, la versión utilizada para las pruebas es Snow Leopard 10.7.

Para la prueba instalamos y ejecutamos las aplicaciones Air desktop de nuestro sistema (QDelivery.air, QEstadistica.air y QMantenimiento.air), demostrando la correcta ejecución de las aplicaciones en esta sistema operativo.

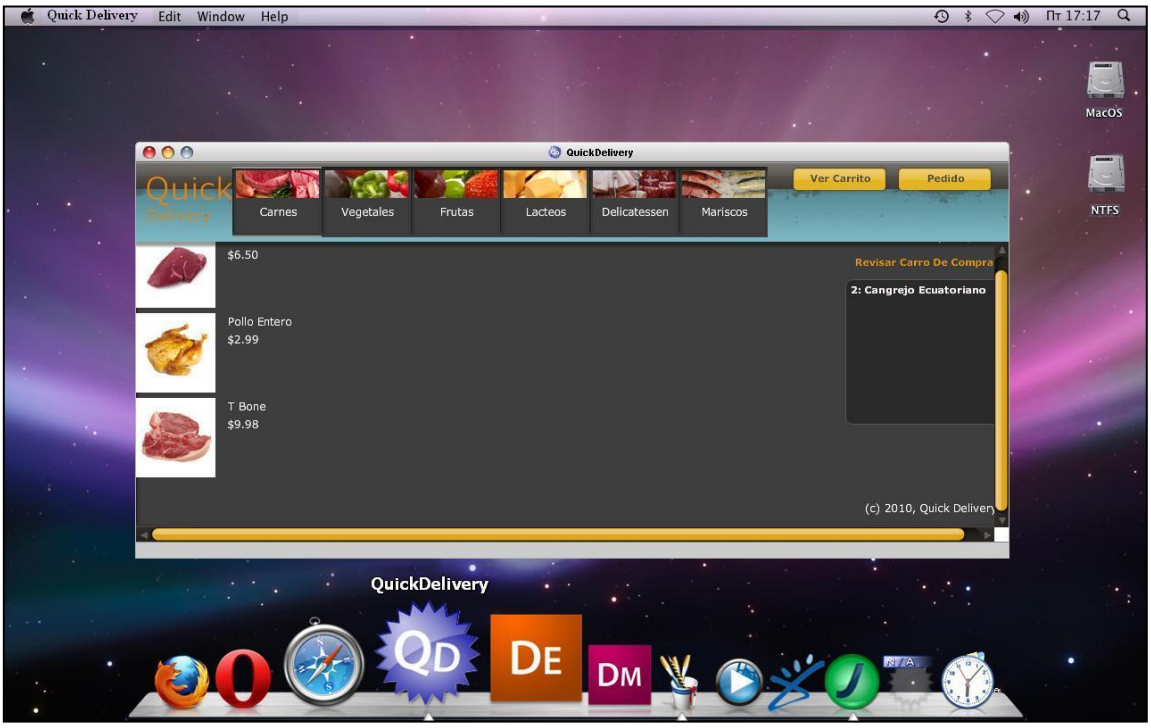


Figura 7.29 Pantalla de la aplicación Air Desktop QDelivery ejecutada en Mac OS Snow Leopard 10.7

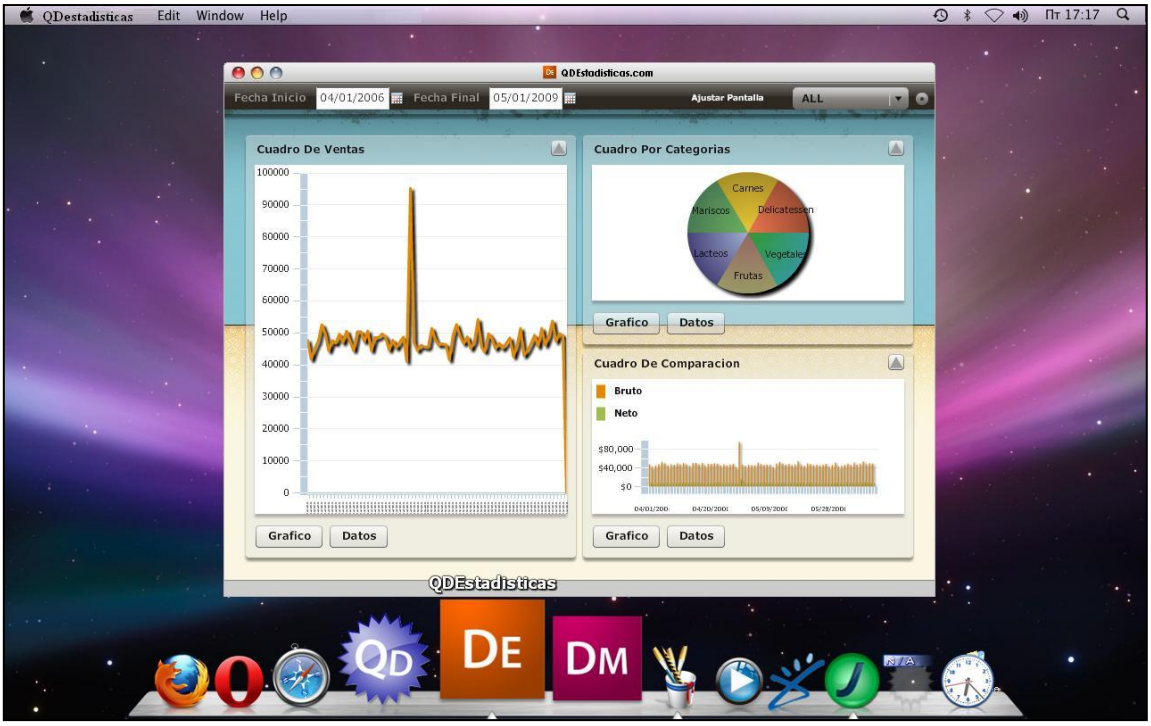


Figura 7. 31 Pantalla de la aplicación air desktop QEstadistica ejecutada en Mac OS Snow Leopard 10.7

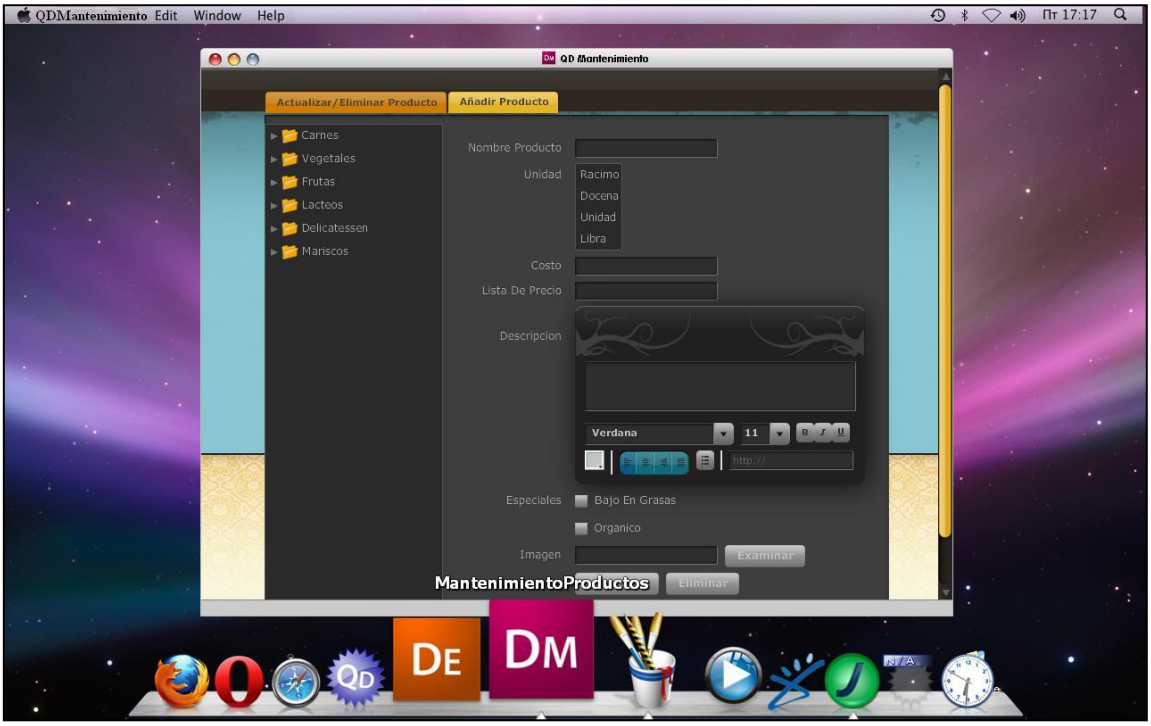


Figura 7. 30 Pantalla de la aplicación air desktop QMantenimiento ejecutada en Mac OS Snow Leopard 10.7

7.6.2.3. Pruebas de función

Se evaluaron las distintas funcionalidades de la aplicación web entre estas las descargas de las aplicaciones Air desktop, el mantenimiento de los productos y el correcto funcionamiento de los cuadros estadísticos de las ventas. Anteriormente se ya se realizó el análisis de estas aplicaciones en cuanto a memoria y rendimiento, ahora las pruebas se basaron en identificar posibles fallas durante sus ejecuciones.

1. **Prueba de descarga**

   Se realizó las descargas de cada una de las aplicaciones Air desktop que se encuentran en el menú principal del sitio web. Ninguna tuvo falla al momento de la descarga, no presentaron error alguno al guardar el ejecutable *.air y se instalaron satisfactoriamente.

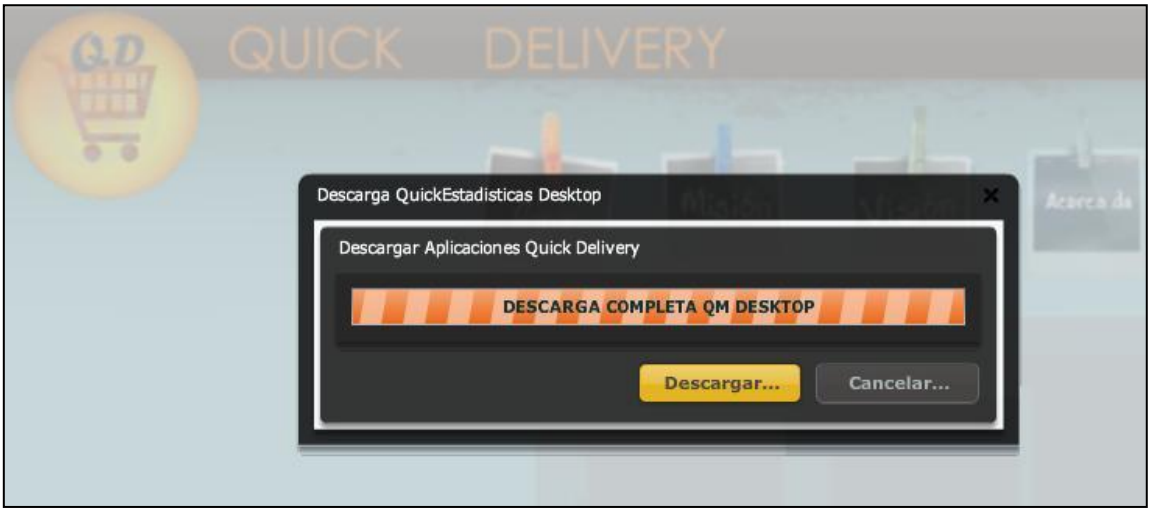


Figura 7. 32 Pantalla de descarga de QEstadisticas.air

2. Prueba de mantenimiento de productos

El ingreso, eliminación y actualización de los productos no tuvo problemas. La base de datos reaccionó correctamente se efectuaron todos los cambios y a su vez la página de Quick Delivery actualizó correctamente los productos nuevos o modificados.

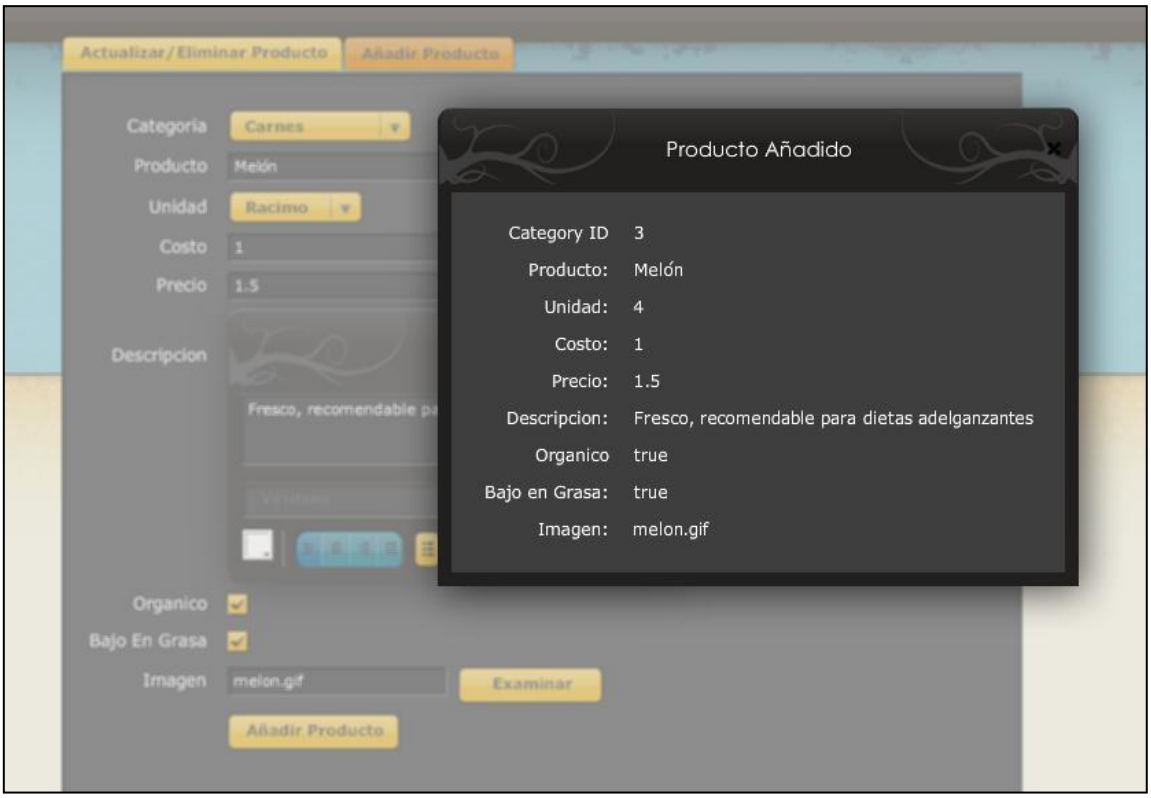


Figura 7. 33 Pantalla de producto añadido en QMantenimiento

3. Prueba de ventas estadísticas

Se evaluaron a los distintos gráficos y cuadros estadísticos, y estos respondieron adecuadamente a los datos generados de todas las ventas, los cambios de fechas, cambios entre categorías y la variación entre gráficos.

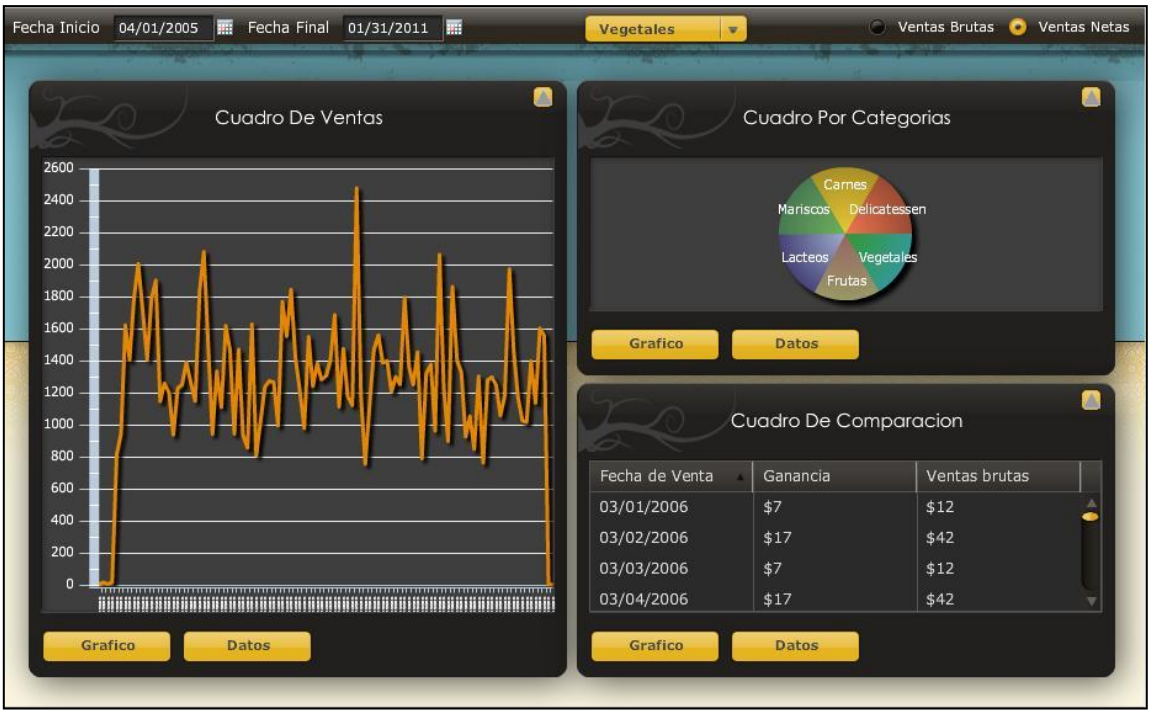


Figura 7. 34 Pantalla de funcionamiento de QEstadistica

7.2.6.4. Pruebas de configuración

Se comprobaron que las aplicaciones del sistema tengan sus correctas configuraciones en distintos hardware y software. Se aseguró el correcto funcionamiento de las opciones y funcionalidades de las instalaciones; verificando que todos los componente necesarios se han instalado. En caso de desinstalar las aplicaciones se constató que todos los datos, DLLs y ejecutables fueron removidos.

Para la instalación de las aplicaciones Air de escritorio es necesario instalar previamente Adobe Air Runtime, que está disponible para todas las plataformas.

Platform	Installation location
Windows 2000, XP, Vista, and Windows 7 [32 bit]	C:\Program Files\Common Files\Adobe AIR\
Windows Vista and Windows 7 [64 bit]	C:\Program Files (x86)\Common Files\Adobe AIR\
Mac OS X	/Library/Frameworks/Adobe AIR.framework/
Linux [32 bit and 64 bit]	/opt/Adobe AIR

Figura 7. 35 Plataformas que soportan Adobe Air

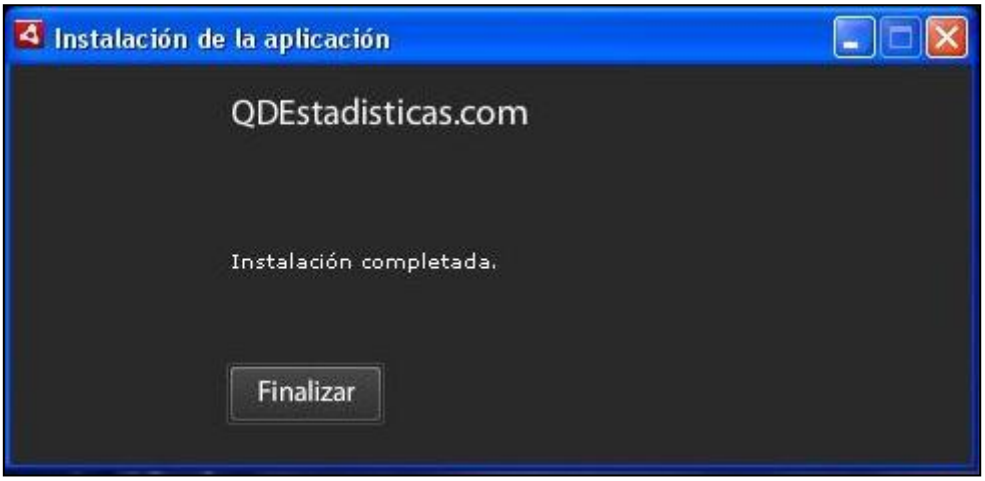


Figura 7. 36 Instalación QEstadisticas.air en Windows

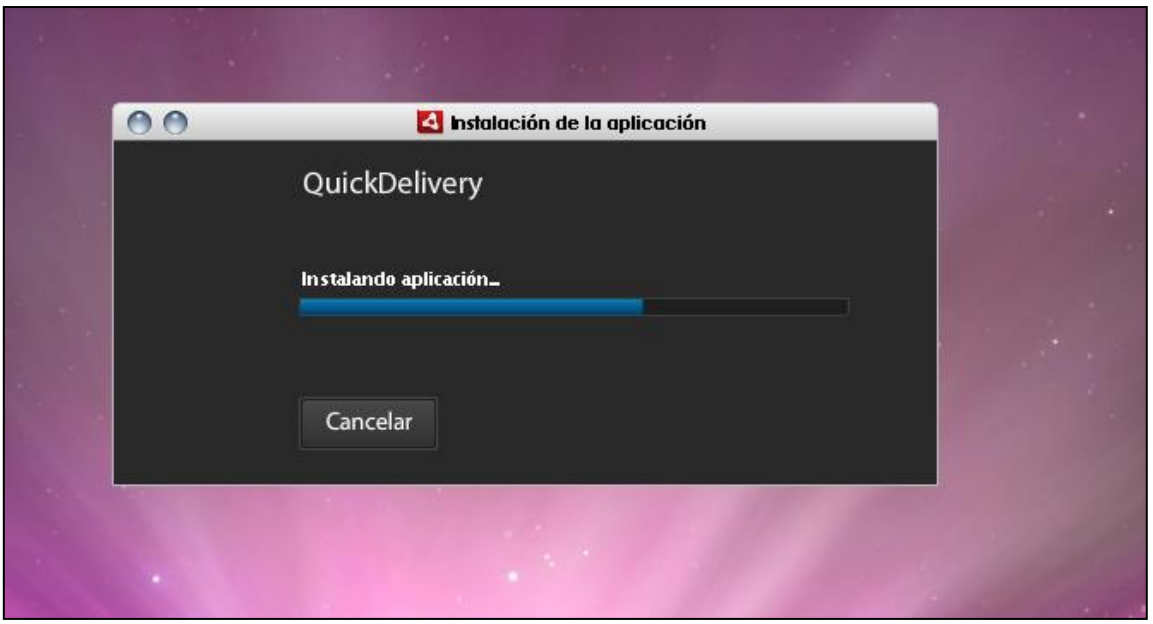


Figura 7. 37 Instalación QDelivery.air en Mac OS

7.2.6.5 Pruebas de navegación

Dentro de una AppWeb es muy importante la navegación dentro de la misma y para esto hemos realizado pruebas de navegación dentro de la aplicación.

Mostraremos qué pantallas puede ver el usuario y cuándo puede verlas y comprobar que pueda elegir por donde puede moverse.

En algunos aspectos de la navegación, el mismo usuario puede decidir cómo hacer click en un botón para moverse desde la página principal al proceso de verificación del pedido o al carrito de compras.

Otros aspectos de la navegación estará controlada por la aplicación; por ejemplo durante el proceso de verificación el usuario no puede pasar a la pantalla siguiente hasta que haya cumplido con los requisitos de llenar los campos requeridos.

1. Navegación dentro de la página de inicio.

Dentro de la página inicial encontramos 2 tipos de menús que permiten al usuario visualizar las características de la AppWeb y otro menú en el cuál dependiendo del tipo de usuario puede ingresar a la aplicación que desee.

Además de tener links de descargas para las aplicaciones air.

Figura 7. 38 Navegación Página De Inicio

2. Navegación dentro de Quick Delivery

El Proceso de verificación del pedido está controlado por viewstacks, que es uno de los contenedores navigators de Flex. El usuario siempre sabrá en qué estado se encuentra.

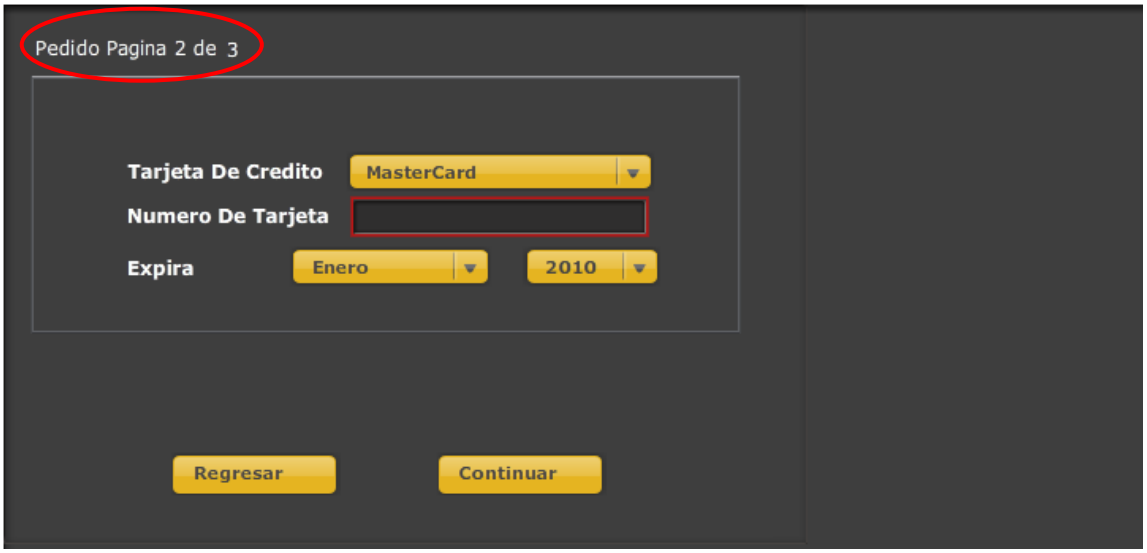


Figura 7. 39 Navegación Quick Delivery

3. Navegación en QMantenimiento

Dentro del mantenimiento de productos podemos movernos a través de TabNavigator para separar la funcionalidad de los dos componentes Actualizar/EliminarProducto y Añadir Producto en dos pestañas diferentes.

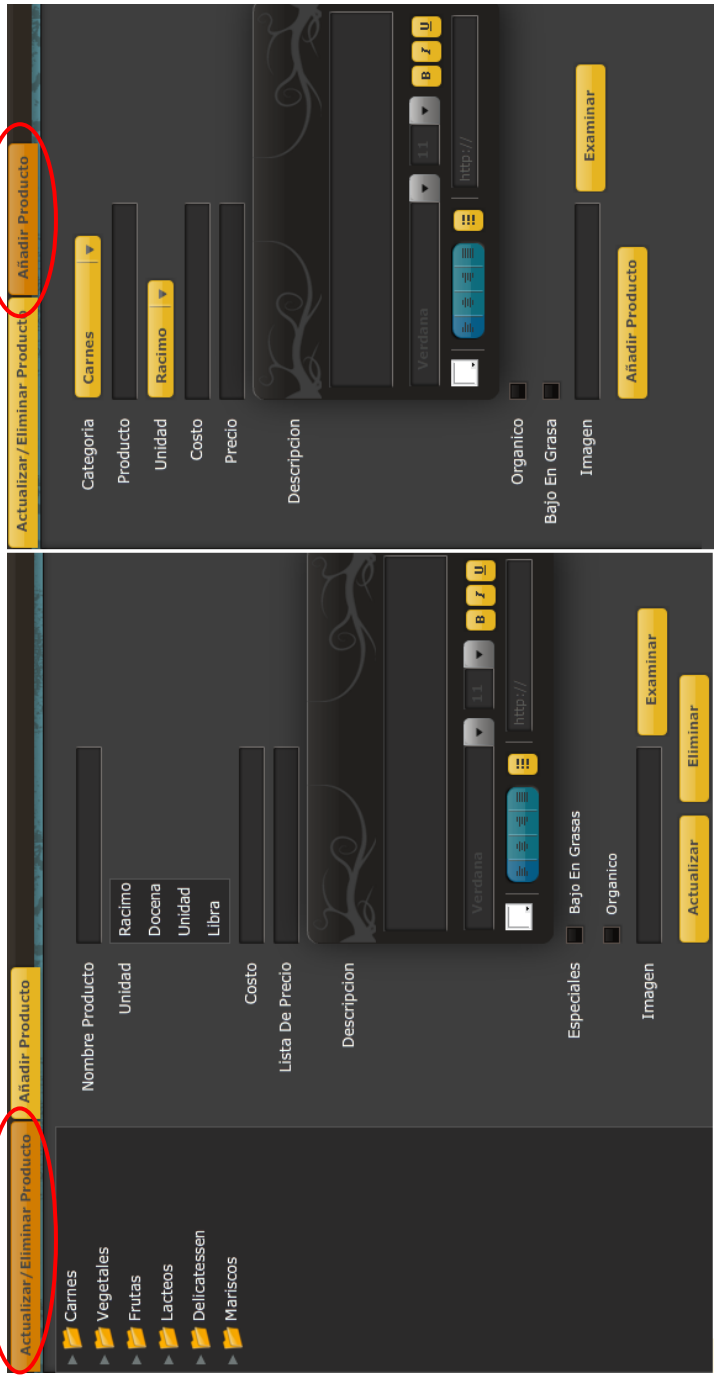


Figura 7. 40 Navegación QMantenimiento

4. Navegación en QEstadísticas

En QEstadísticas los distintos gráficos estarán todos expuestos. En caso de que el usuario requiera visualizar distintas categorías tiene una lista desplegable, también puede moverse entre tablas con datos y los gráficos a través de LinkButtons.

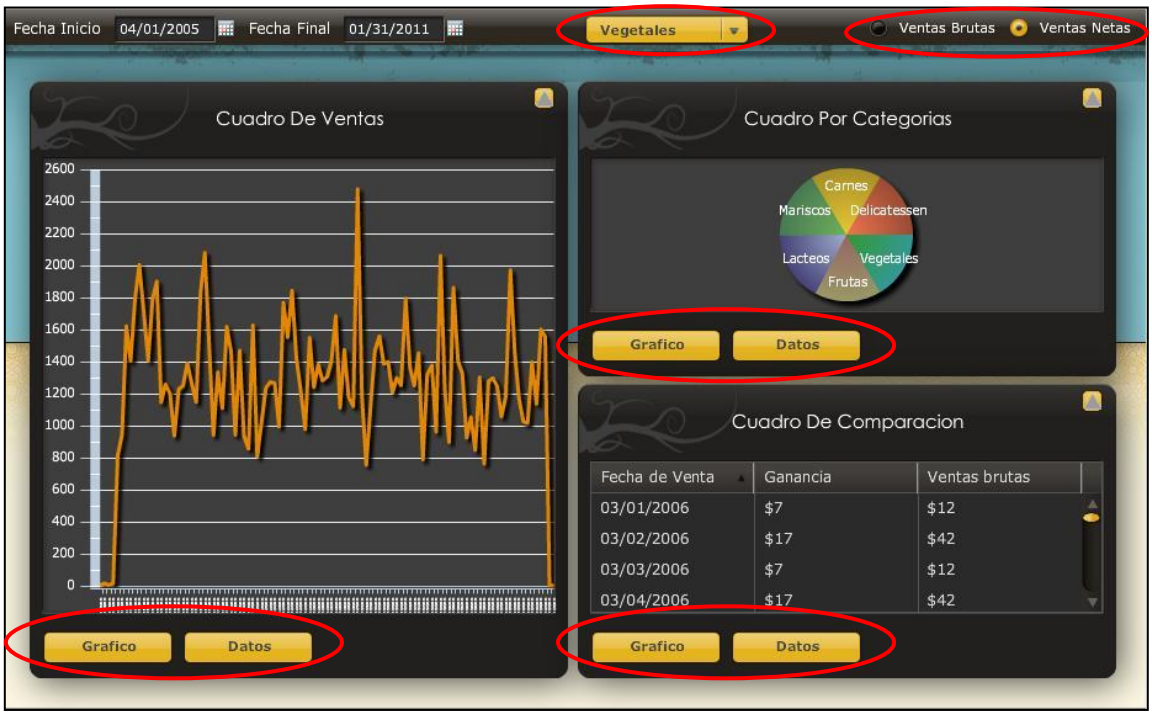


Figura 7. 41 Navegación QEstadísticas

7.3. Conclusión

En este capítulo hemos ejecutado diferentes tipos de pruebas y diagnósticos que nos ayudaron a identificar en que parte de un la estructura del sistemas Quick Delivery se produce fuga de memoria o inconsistencia de datos, los mismos que representan un gran riesgo en la integridad funcional de nuestra aplicación.

Usando una herramienta llamada Profiler de Flex Builder 3, hemos podido realizar una evaluación sobre los diferentes componentes de cada módulo, identificando algunos de los riesgos a los que nuestro proyecto Web está expuesto, a la vez hemos analizado el estado de la memoria del equipo que alberga nuestro sistema determinando cual será el posible tratamiento que se le debe dar a cada uno de los problemas encontrados.

Entre las pruebas que hemos realizado podemos mencionar las siguientes:

- Pruebas utilizando Profiler Test para detectar fugas de memoria en la Web App.
- Pruebas de rendimiento usando Perfomance Profiler.
- Pruebas de interfaz de usuario.
- Pruebas de compatibilidad (Navegadores y Sistemas Operativos).
- Pruebas de funcionalidad.
- Pruebas de configuración.
- Pruebas de navegación.

Además, en este capítulo hemos utilizado herramientas de alta tecnología para realizar la identificación de los errores que causan pérdida de memoria y repercuten en el rendimiento de la aplicación, llegando a identificar entre los posibles causantes de este daño:

- Asignación de la memoria por parte de flash Player.

- Recolección de elementos no utilizados en Flash Player.
- Fugas ocasionadas por los escuchadores de eventos.

Los mismos puntos que hemos tratado debidamente dando solución a cada uno de estos, dejando a punto todos los componentes del Sistema que estamos desarrollando.

CONCLUSIONES

Al finalizar el desarrollo de nuestra tesis y habiendo cumplido con todos los objetivos planteados en el diseño de esta, hemos obtenido grandes conclusiones sobre las tecnologías RIA´s, de lo que resaltaremos el potencial y la rapidez que aporta en la creación de sitios Web Dinámicos y llamativos para los usuarios gracias al uso de herramientas como Adobe Flex Builder 3, la versatilidad al momento de implementar nuestros proyectos en los diferentes sistemas operativos y dispositivos de hardware, haciendo uso del plugin de Adobe AIR que se instala coherentemente tanto en el navegador como en el escritorio.

Mediante el análisis que realizamos en base a los requisitos obtenidos, pudimos construir diagramas y modelos como casos de uso, diagrama entidad-relación, diagramas de actividad entre otros, con lo que se pudo entender y construir nuestro sistema implementando la arquitectura MVC, para lograr realizar cambios en una parte estructural de la aplicación con un impacto mínimo en las demás partes, ayudándonos mediante la creación de aplicaciones modulares con la tecnología de Flex. Además aplicamos el concepto de RSL´s (Runtime Shared Libraries), con lo que no solo dividimos partes de la aplicación en módulos sino que creamos librerías de código común, para compartir código a lo largo de varias aplicaciones reduciendo el tamaño de los archivos de esta aplicación.

El uso de ColdFusion como servidor de aplicaciones nos da la lógica estructural para consumir los Web Services, los mismos que usan XML para indicar cómo se va a estructurar el formato de la información generada por QuickDelivery.

La parte visual del Sitio Web fue desarrollada gracias al análisis del diseño de la interfaz que es muy importante para que los usuarios interactúen con él sistema. Al identificar los riesgos a los que se expone nuestro proyecto pudimos elaborar un plan de contingencia con el que aseguramos el cumplimiento de los objetivos planteados.

Llevamos a cabo la codificación de nuestro sistema, a la par que aprendimos a trabajar con la herramienta Adobe Flex. El proyecto que creamos lleva el nombre de QuickDelivery y se estructura de tres partes las cuales son QDMantenimientos,

QDEstadisticas y QuickDelivery que componen un sistema de Comercio Electrónico de venta de productos alimenticios con una parte que se ocupa de mantenimientos en general y otra parte orientada a mostrar cuadros estadístico de las ventas realizadas.

Además acoplamos funcionamiento de otras librerías de código abierto como PaperVision para modelamiento de un formulario en 3D, Google Maps y Google Languaje que aportan de grandes utilitarios a nuestro sistema.

En lo que respecta a las pruebas, hemos utilizado herramientas de punta en el desarrollo de RIA´s para realizar la identificación de los errores que causan pérdida memoria y repercuten en el rendimiento de la aplicación, los mismos puntos que hemos corregido con prontitud.

El proyecto QuickDelivery haciendo uso de estos métodos y herramientas descritas, está disponible tanto Online como Offline mediante la descarga de cualquiera de sus tres aplicaciones, las mismas que son totalmente compatibles con cualquier Sistema Operativo y dispositivo de hardware.

RECOMENDACIONES

Luego de haber concluido el desarrollo de este trabajo investigativo, nos permitiremos presentar las siguientes recomendaciones:

Realizar el análisis y levantamiento de requerimientos antes de proceder con la programación del sistema para evitar contratiempos por falta de planeación.

Escoger las herramientas de acuerdo al alcance que estimemos que tendrá nuestro proyecto informático para evitar demoras y gastos innecesarios.

Usar características como modularidad y librerías de código para aprovechar mejor el tiempo de desarrollo y reutilizar recursos ahorrando tiempo y dinero durante este proceso.

Realizar las pruebas que se estimen necesarias incluyendo pruebas de rendimiento y uso de recursos del computador para localizar posibles fugas de memoria y problemas que causen mal funcionamiento de nuestro software.

En el software que desarrollemos proponer una visión no restrictiva y de libre aplicación usando herramientas que se acoplen coherentemente en variedad de equipos electrónicos como computadores, smartphones, tablets entre algunos, también permitir la libre instalación y ejecución en múltiples sistemas operativos para brindar flexibilidad, portabilidad y libertad de uso a los diferentes usuarios que nuestro sistema llegue a tener.

GLOSARIO

ActionScript: Lenguaje de programación orientado a objetos (OOP), utilizado en especial en aplicaciones web animadas.

Adobe AIR: Permite a los desarrolladores utilizar HTML, JavaScript, Adobe Flash y ActionScript para crear aplicaciones web que se ejecutan como aplicaciones clientes independientes sin las restricciones de un explorador.

Adobe Flex Builder: Es una herramienta de desarrollo basada en Eclipse muy productiva que incorpora las siguientes funciones: códigos inteligentes, depuración interactiva estratificada, además del diseño visual del aspecto y comportamiento de la interfaz de usuario de las aplicaciones de Internet sofisticadas (RIA).

AJAX: Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*).

API: Es una interfaz de programación de aplicaciones (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

Applets: *Es un* componente de una *aplicación* que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

Array: Una matriz o vector (llamados en inglés arrays) es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, los elementos de la matriz.

AWT: La **Abstract Window Toolkit** (AWT, en español Kit de Herramientas de Ventana Abstracta) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java.

CGI: Interfaz de entrada común (*Common Gateway Interface*) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa.

Componente: Los componentes de Software son todos aquellos recursos desarrollados para un fin concreto y que puede formar solo o junto con otros, un entorno funcional requerido por cualquier proceso predefinido. Son independientes entre ellos, y tienen su propia estructura e implementación.

Cross-domain policy: Es un documento XML que otorga a un cliente web (como Adobe Flash Player, Adobe Reader, etc.) un permiso para manejar los datos a través de dominios múltiples.

CrossPlataform: Multi-plataforma. Todo software o dispositivo es capaz de poder utilizarse en diferentes plataformas, como ser, diversos sistemas operativos.

CSS: Las hojas de estilo en cascada (*Cascading Style Sheets*), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Deep Linking: Enlace profundo, concepto que describe que las aplicaciones que son despachadas con un único URL también puedan admitir navegación basada en URL.

DOM: El Document Object Model ('Modelo de Objetos del Documento') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

Drag & drop: Arrastrar y soltar es una expresión informática que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana. Los objetos arrastrados son habitualmente archivos, pero también pueden ser arrastrados otros tipos de elementos en función del programa.

E4X: La especificación de ECMAScript for XML define un conjunto de clases y funcionalidad para trabajar con datos XML. Este conjunto de clases y funcionalidades se denomina *E4X*.

Eclipse: Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

ECMAScript: Define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo de C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases.

Firewall: Un cortafuegos (*firewall)* es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas

Flash Player: Es una aplicación en forma de reproductor multimedia distribuido por Adobe Systems. Permite reproducir archivos SWF que pueden ser creados con la herramienta de autoría Adobe Flash, con Adobe Flex o con otras herramientas de Adobe y de terceros. Estos archivos se reproducen en un entorno determinado (en un sistema operativo tiene el formato de aplicación del sistema, mientras que si el entorno es un navegador, su formato es el de un Plug-in u objeto ActiveX).

Gadgets: Es un dispositivo que tiene un propósito y una función específica, generalmente de pequeñas proporciones, práctico y a la vez novedoso. Los gadgets suelen tener un diseño más ingenioso que el de la tecnología corriente

Html: Siglas de HyperText Markup Language (*Lenguaje de Marcado de Hipertexto*), es el lenguaje de marcado predominante para la elaboración de páginas web.

Http: Hypertext Transfer Protocol o HTTP (en español *protocolo de transferencia de hipertexto*) es el protocolo usado en cada transacción de la World Wide Web. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IDE: Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Loosely coupled: Se utilizan para construir aplicaciones más fáciles de mantener que reducen el riesgo de cambio de una parte de la aplicación que forzaría un cambio en otra. Con esto nos referimos a la construcción de una aplicación *loosely coupled* (débilmente acoplada).

MVC: Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

PaperVision: Papervision permite hacer elementos en 3D, es un motor de gráficos para Flash que se implementa a través de librerías en cualquier aaplicación.

PHP: Del inglés **h***ypertext* **p***reprocessor* (acrónimo recursivo). Un lenguaje de programación utilizado mayormente para desarrollar servicios web. PHP es un lenguaje de fácil aprendizaje, distribuido en forma gratuita, que permite interactuar con muchos sistemas de gestión de bases de datos.

RCP: Es el acrónimo de Rich Client Plattform. Es la arquitectura sobre la que funciona Eclipse y que a su vez nos permite ampliar y extender la funcionalidad de uno de los IDE´s de Java más populares.

RIA: Acrónimo de ***R****ich* ***I****nternet* ***A****pplications.* Son aplicaciones web que tienen la mayoría de las características de las aplicaciones tradicionales, estas aplicaciones utilizan un "navegador web" estandarizado para ejecutarse y por medio de "plugin" o independientemente una "virtual machine", se agregan las características adicionales. Buscan mejorar la experiencia del usuario.

Runtime Shared Library: RSL es un mecanismo para reducir el tamaño de sus aplicaciones y reducir así el tiempo necesario para descargar la aplicación. RSL es un archivo independiente que el cliente descarga por separado desde el archivo SWF de la aplicación, y almacena en caché en el equipo cliente para su

uso con aplicaciones múltiples archivos SWF. El uso de un RSL reduce el tamaño del archivo resultante para sus aplicaciones.

Servidor Apache: Es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP.

Servidor De Aplicaciones: Se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones. Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Los principales beneficios son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Silverlight: Microsoft Silverlight es una estructura para aplicaciones web que agrega nuevas funciones multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones e interactividad, en forma similar a lo que hace Adobe Flash.

SOA: La **A**rquitectura **O**rientada a **S**ervicios de cliente, es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

UDDI: O también llamado catálogo de negocios de Internet. Para el registro en este catálogo se deberá de usar XML. Es una iniciativa industrial abierta basada en el concepto de los Web Services.

Web Dinámica: Se conoce con el nombre de página web dinámica a aquélla, cuyo contenido se genera a partir de lo que un usuario introduce en un web o formulario. El contenido de la página no está incluido en un archivo html como en el caso de las páginas web estáticas.

Web Service: Un servicio web es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas

sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet

WPF: Windows Presentation Foundation permite el desarrollo de interfaces de interacción en Windows tomando las mejores características de las aplicaciones Windows y de las aplicaciones web. Ofrece una amplia infraestructura y potencia gráfica con la que es posible desarrollar aplicaciones visualmente atractivas, con facilidades de interacción que incluyen animación, vídeo, audio, documentos, navegación o gráficos 3D.

XAML: Acrónimo de e***X****tensible* ***A****pplication* ***M****arkup* ***L****anguage* AML es un lenguaje declarativo basado en XML, optimizado para describir gráficamente interfaces de usuarios visuales ricas desde el punto de vista gráfico, tales como las creadas por medio de Adobe Flash.

XML: Siglas de e***X****tensible* ***M****arkup* ***L****anguage* ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

BIBLIOGRAFÍA

Libros:

- ✓ **Adobe Air 1.5 CookBook**

  Tucher David, Casario Marco, Koen De Weggheleire y Rich Tretola.

  Adobe, 2009

- ✓ **Adobe Flex 3**

  Tapper Jeff, Labriola Michael, Mathew Boles Y Talbot James.

  Adobe, 2008

- ✓ **Action Script Programación Interactiva Al Máximo**

  De La Cruz Villar Joel

  Megabyte, Junio 2004

- ✓ **Comercio Electrónico**

  Alarcón Herrera Erika, Crovetto Huerta Christian

  Megabyte, Enero 2005

Páginas Web:

- ✓ Adobe Systems Incorporated. Adobe Flex 3.

  http://livedocs.adobe.com/flex/3/html/, 2010.

- ✓ Adobe Systems Incorporated. Flex 3 and ColdFusion.

  http://learn.adobe.com/wiki/display/Flex/Flex+and+ColdFusion, 2007.

- ✓ Adobe Systems Incorporated. Web Services.

  http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=webservices_02.html, 2007

- ✓ Enrique Chávez. Rich Internet application

  http://www.riactive.com/2006/12/06/rich-internet-application-definicion/, Diciembre 2006.

- ✓ Florian Moritz. Rich internet applications (ria). a convergence of user interface paradigms of web and desktop, 2008. Diploma Thesis. University of Applied Science Kaiserslautern.

  http://www.flomedia.de/diploma

- ✓ Google. Google Maps API for Flash.

  http://code.google.com/intl/es-ES/apis/maps/documentation/flash/, 2010.

- ✓ Jayaram Krishnaswamy, Creating a Web Service with ColdFusion: the Basics

  http://www.devarticles.com/c/a/ColdFusion/Creating-a-Web-Service-with-ColdFusion-the-Basics/3/, Febrero 2006.

- ✓ Jesse James Garrett. Ajax: Un Nuevo enfoque de las aplicaciones web (Ajax: A new approach to web applications), Febrero 2005.

  http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

- ✓ Marcio Gallio, Roger Soares, y Ian Oeschger. Inner-browsing: Extendingweb browsing the navigation paradigm,2003

  http://devdgetemp.mozilla.org/viewsource/2003/inner-browsing/index-en.html,

- ✓ O'Reilly, Programming ColdFusion MX: Web Services.

  http://www.webreference.com/programming/coldfusion/1/,Agosto 2003.

- ✓ Serrano Cinca C. "El Comercio Electrónico en los departamentos de una empresa".

  http://www.ciberconta.unizar.es/leccion/econta/inicio.html, 2011.

- ✓ Tim Berners-Lee. Information management: A proposal,

  http://www.w3.org/History/1989/proposal.html, 1989.

- ✓ Tim O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software.

  http://www.oreilly.com/, September 2005.

ANEXOS

ANEXO 1

Entrevista

Tema: Servicio De Entregas A Domicilio

Lugar y Dirección: Remigio Crespo 3-24 y Agustín Cueva

Fecha y Hora: Miércoles 17 de octubre de 2009, 9:10 am

Duración: 40 minutos

Nombre del Entrevistado: Darshan Montesinos Arriaga

Objetivos

- Conocer el funcionamiento de la empresa.
- Identificar cómo se realizan las entregas de los productos al usuario final.
- Conocer si una página web realizada con tecnología RIA mejoraría sus ventas y sistema.

Preguntas

A continuación las preguntas realizadas en la entrevista.

1. **¿Cuánto tiempo de existencia tiene la empresa?**

   La empresa está recién en sus inicios, tiene apenas 2 años 10 meses de existencia.

2. **¿Cuál es el objetivo de su empresa?**

   El principal objetivo es entregar los productos a los diferentes clientes en su domicilio de manera eficiente, actuando como una tercerizadora entre el cliente y el proveedor.

3. **¿Qué tipo productos se entregan?**

Entre los principales se encuentra la comida, en la cual se ofrece heladerías, bebidas, restaurantes y dulcerías, pero la propuesta se puede ampliar para los diferentes proveedores que quieran unirse a este servicio.

4. **¿Tienen distinción entre clientes?**

   Si, a los clientes frecuentes se les puede dar la opción de pagar en cheque o se les puede dar a crédito los pedidos.

5. **¿Cuáles son las formas de pago?**

   Por el momento, los clientes tienen que pagar en efectivo, pero se está pensando en la idea de manejar tarjetas de crédito, por lo cual están esperando que el Municipio nos entregue la patente para poder realizarlo.

6. **¿De qué manera entregan los productos?**

   Utiliza como mensajeros predefinidos a motociclistas quienes tienen un sueldo mensual, pero en el caso de requerir auxiliares disponen de otras personas que les facilitan la entrega siendo estas pagadas al instante.

7. **¿Tiene recargo adicional el costo de los productos?**

   No, el único recargo es el costo de la entrega, pero esta empresa posee un convenio con sus proveedores quienes les dan un cierto porcentaje de descuento por producto.

8. **¿Si el pedido se encuentra fuera del área de cobertura, existe un recargo adicional al costo de envío?**

   Con respecto al área urbana, se manejan tres zonas las cuales tienen diferentes precios que son de $1.50, $2.00 y $2.50, pero en el área rural el mínimo es de $3.50

9. **¿Cómo se encuentran distribuidos los repartidores?**

   No se encuentran distribuidos, debido a que se encuentran en la central y de allí parten para entregar los pedidos.

10. **¿Cuál es el horario de atención?**

    Depende del horario de atención de nuestros proveedores.

11. **¿Los pedidos se realizan por vía telefónica, o poseen otro medio para realizarlo?**

    A más de realizarlo mediante vía telefónica, se pueden realizar los pedidos mediante el Chat, pero están pensando en realizarlo vía celular.

12. **¿Cómo realizan los empleados las entregas?**

    Por cuestiones de tiempo, cada empleado se encarga de un pedido a la vez.

13. **¿Cree que una página web dinámica facilitaría y mejoraría las ventas de su empresa?**

    Claro que sí, sería mucho más fácil si nuestros clientes conocieran mejor nuestros productos mediante una página web donde ellos puedan a su gusto buscar y escoger lo que quisieran, a la hora que quieran, desde cualquier lugar y tomándose su tiempo para hacer el pedido, porque a muchos de los clientes les interesa la rapidez y la facilidad con la que puedan comprar.

ANEXO 2

Instalación de adobe AIR

1.- Descargamos el instalador de Adobe Air para el sistema operativo que deseemos desde http://get.adobe.com/es/air/otherversions/

2.- Hacemos doble clic en el archivo AIR descargado con su extensión respectiva según el sistema operativo a utilizarse y se abrirá una ventana de instalación.

3.- completada la instalación hacemos clic en OK.

Instalación en MAC OS Snow Leopard 10.7

1. Hacemos doble clic en AIR.pkg se abrirá la ventana de Install Adobe Integrate Runtime 1.0.

2. Hacemos clic en el botón continue en la esquina inferior izquierda de la ventana. Se abrirá el cuadro de diálogo Select a Destination page.

3. Seleccionamos el volumen de destino y hacemos clic en continue. Se abrirá el botón Easy Install.

4. Hacemos clic en el botón Install en la esquina inferior izquierda de la ventana. El instalador muestra una ventana Authenticate, y escribimos el nombre del usuario y la contraseña de la MacOS.

5. Cuando termine la instalación cerramos usando el botón de Close.

Para la instalación en Windows o Linux los pasos son los mismos.

ANEXO 3

CÓDIGO DE LA IMPLEMENTACION API GOOGLE MAPS EN EL PROYECTO QUICK DELIVERY

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application backgroundImage="assets/images/application_bg.png"
xmlns:mx="http://www.adobe.com/2006/mxml" xmlns:maps="com.google.maps.*"
  width="100%" height="100%" layout="absolute" viewSourceURL="srcview/index.html">
 <mx:Panel title="Google Maps Geocoding JDCompusystem" width="1047" height="458"
horizontalCenter="0" verticalCenter="9">

    <maps:Map
    id="map"      sensor="false" key="your_api_key"
    mapevent_mapready="onMapReady(event)"
    width="100%" height="100%"/>
            </mx:VBox>
 </mx:Panel>
 <mx:Script>
    <![CDATA[
      import com.google.maps.services.ClientGeocoderOptions;
      import com.google.maps.LatLng;
      import com.google.maps.Map;
      import com.google.maps.MapEvent;
      import com.google.maps.MapMouseEvent;

      private function doGeocode(event:Event):void {
        // Geocoding example
        var geocoder:ClientGeocoder = new ClientGeocoder();

        geocoder.addEventListener(
          GeocodingEvent.GEOCODING_SUCCESS,
          function(event:GeocodingEvent):void {
          var placemarks:Array = event.response.placemarks;
          if (placemarks.length > 0) {
            map.setCenter(placemarks[0].point);
            var marker:Marker = new Marker(placemarks[0].point);

            marker.addEventListener(MapMouseEvent.CLICK, function (event:MapMouseEvent):void {
               marker.openInfoWindow(new InfoWindowOptions({content: placemarks[0].address}));
            });
            map.addOverlay(marker);
          }
        });
        geocoder.addEventListener(
          GeocodingEvent.GEOCODING_FAILURE,
          function(event:GeocodingEvent):void {
            Alert.show("Geocoding failed");
            trace(event);            trace(event.status);
          });        geocoder.geocode(address.text);
      }
    ]]>
 </mx:Script> </mx:Application>
```

ANEXO 4

Denuncia De Tesis

SR.DR.ROMEL MACHADO CLAVIJO SECRETARIO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION

CERTIFICO .Que, El H. Consejo de Facultad en sesión del 26 de Junio del 2009 conocio el informe dela Junta Academica de la escuela de Ingenieria de Sistemas en base a esta ,aprobo la denuncia de la Tesis presentada por los señores Daysi Becerra Naranjo y Juan Andres Velez con el.. tema APLICACIÓN PRACTICA DE RIA (RICH INTERNET APPLICATIONS) EN EL COMERCIO ELECTRONICO CON EL DESARROLLO DE UN SISTEMA DE ENTREGAS Y REPARTOS DELIVERY Se. ratifica como Director al Ingeniero Pablo Pintado Sumba y como miembros del Tribunal a los señores Profesores Ingenieros Pablo Esquivel Leon y Patricia Ortega Chasi los denunciantes tienen un plazo minimo de cuatro meses y un maximo de diez y ocho meses a partir de la fecha de aprobación es decir hasta el 26 Diciembre del 2010.

Cuenca, 6 de Julio del 2009

UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACION

ESCUELA DE INGENIERIA DE SISTEMAS

DISEÑO DE TESIS

TEMA:

"Aplicación práctica de RIA (Rich Internet Applications) en el comercio electrónico con el desarrollo de un Sistema de Entregas y Repartos DELIVERY"

AUTORES:
DAYSI BECERRA NARANJO.
JUAN ANDRES VELEZ.

DIRECTOR DE TESIS:

ING. PABLO PINTADO

CUENCA, ECUADOR
2009

1. **SELECCION Y DELIMITACION DEL TEMA**

"Aplicación práctica de RIA (Rich Internet Applications) en el comercio electrónico con el desarrollo de un Sistemas de Entregas y Repartos DELIVERY"

2. **RESUMEN**

En la actualidad todos hemos experimentado los beneficios que nos provee el uso del Internet tanto con fines académicos, comerciales, sociales y de comunicación en general; llegando a convertirse más que en un elemento de entretenimiento en una herramienta de trabajo y desarrollo de los pueblos alrededor del mundo. En la actualidad el número de personas que accede al Internet crece de manera exponencial cada día, y también sus exigencias sobre los servicios y beneficios que sobre este pueden obtener, y es así como los actuales navegadores como Internet Explorer, Mozilla Firefox entre otros, se encuentran cada vez más exigidos frente al desarrollo de Internet y a la enorme cantidad de aplicaciones que se han sumado a la red en los últimos años. Música en línea, la posibilidad de cargar imágenes y videos en blogs y espacios de redes sociales, sitios de almacenamiento virtual, páginas con presentaciones multimedia, documentos de diferente tipo que se editan online en forma compartida, como en el caso de Google Docs, todas estas aplicaciones tornan más "pesada" y compleja a la web y, en consecuencia, ponen a los navegadores actuales en el límite de sus posibilidades.

Ante esto, han surgido herramientas web que buscan concretar un paso adelante tecnológicamente hablando, lo que permitiría una actividad más relajada en Internet. Se llaman RIA (Rich Internet Applications) y permiten descargar en un único paso toda la información necesaria, sin el intercambio con los servidores que caracteriza a la tecnología actual. En consecuencia, la navegación y las tareas en la web se vuelven más fluidas.

Para llevar a la práctica lo antes descrito con la tecnología RIA, se ha pensado en el desarrollo de un sistema Delivery que se preocupa del diseño, planificación, implementación y mejoramiento de los flujos asociados a la entrega, generalmente sujeta a restricciones de tiempos y costos. El resultado de la gestión del reparto o Delivery es la operación de colocar en el tiempo acordado, en las condiciones acordadas, y a la persona adecuada la cantidad precisa de un bien o servicio demandado.

3. CONTEXTUALIZACION

El comercio electrónico se ha convertido en poco tiempo en la herramienta tecnológica más revolucionaria de todas, influyendo en prácticamente todos los niveles de la actividad humana, ya que nos genera un nuevo canal de ventas con el cual se puede ampliar y expandir una empresa llegando a los rincones más alejados del planeta, lo cual nos facilita la promoción de productos y servicios las 24 horas e incrementar la comunicación con los clientes (Andrew Grove, Estrategia De Negocios Electrónicos).

Tomando en cuenta este antecedente y que día a día va creciendo el número de usuarios en internet y también sus requerimientos y necesidades, se puede decir que las RIA son la nueva generación de las aplicaciones en internet, ya que ofrecen una experiencia sofisticada y atractiva que mejora la satisfacción del usuario y aumenta su productividad. Gracias al amplio alcance de Internet, las RIA pueden implementarse en navegadores, escritorios y dispositivos móviles.

En los entornos RIA, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos.

Las capacidades multimedia son totales gracias a que estos entornos tienen reproductores internos y no hace falta ningún reproductor del Sistema Operativo por parte del usuario.

Para crear este tipo de aplicaciones existen herramientas como Adobe Flex, que es un marco de trabajo de código abierto gratuito que trabaja en conjunto con Adobe Air, las cuales se ejecutan en diferentes Sistemas Operativos permitiendo la fácil distribución de las aplicaciones creadas con estas herramientas.
Gracias al nuevo tiempo de ejecución de adobe Air los desarrolladores podemos utilizar la tecnología Flex para crear aplicaciones de Internet sofisticadas que se implanten en el escritorio.

4. OBJETIVOS

4.1 Objetivo General.

Investigar e implementar una aplicación basada en RIA (Rich Internet Applications) para el comercio electrónico con la creación de un Sistema de entregas y repartos (Delivery)", cuyo uso está dirigido a la ciudadanía Cuencana.

4.2 Objetivos Específicos.

- Investigar sobre cuales son y que ofrecen las nuevas tecnologías y herramientas de desarrollo de aplicaciones en la WEB.
- Profundizar sobre las RIA (Rich Internet Applications), la nueva tendencia en creación de sitios WEB.
- Aprender el uso y las bondades de las herramientas Adobe Air y Adobe Flex para el desarrollo de Aplicaciones RIA.
- Desarrollar un sistema de comercio electrónico para el manejo de entregas y repartos "Delivery", utilizando las Aplicaciones de Internet Enriquecidas.
- Gestionar el mantenimiento de usuarios y administrador del sistema.
- Procesar transacciones de entregas y repartos mediante una página web.
- Actualizar la base de datos de la aplicación de escritorio cuando esta se conecte a internet.
- Ejecutar la transacción cuando la aplicación de escritorio detecte una conexión a internet.
- Realizar consultas a nivel de aplicación aunque esta no esté conectada a internet.

5. IMPACTO TECNOLÓGICO Y SOCIAL

5.1 Tecnológico

- Cada usuario ya no tendrá que esperar por cargar una y otra vez una página con contenido que ahora lo tendrá en el escritorio y no en un servidor ajeno.
- Instalar aplicaciones web en el escritorio.
- Mejora de las capacidades multimedia.
- Manejar grandes volúmenes de información con una sola carga.
- Mejora de las interfaces de usuario, mediante el uso de multimedia y recursos propios de la computadora del usuario.
- Incorporación de múltiples tecnologías en una sola herramienta para facilitar el trabajo de los desarrolladores.
- La aplicación se conectara en internet solo para actualizar las bases de datos.
- La información será guardada de una manera segura con el fin de que no pueda ser extraviada.
- Posibilidad de fusionar el sistema con un servidor de mapas a nivel mundial que nos permita una fácil localización de los clientes y proveedores.
- Permitir a los clientes hacer pedidos de sus productos desde su lugar de residencia o trabajo.
- Seguridad y agilidad al momento de realizar las transacciones.
- Afianzar al usuario a usar las tecnologías de la información para realizar sus transacciones diarias.
- Construir interfaces fluidas" y más usables que imiten la inmediatez del escritorio.
- Las aplicaciones RIA permiten mejores configuraciones, también permiten a las empresas incorporar vídeo y otros contenidos de ayuda contextual a las aplicaciones.

5.2 Social

- Permite que el usuario personalice su pedido al tener los lugares y artículos que estos ofrecen.
- La información va a estar disponible tanto para usuarios como para visitantes del sitio.
- No requiere de una conexión permanente al internet para armar su pedido lo que beneficia a usuarios que pagan por evento su conexión a internet.
- Gracias a su compatibilidad con servidores de mapas permite una fácil localización de los sitios de entrega y abastecimiento.

- Índices y estadísticas de los clientes y proveedores más solicitados.
- Ocultar información que el cliente considere confidencial.
- Permitir el desarrollo de nuevos puestos de trajo.
- Al tener la aplicación en el escritorio se logra economizar en acceso al internet reduciendo costos de navegación que se paga por evento.

6. SITUACION ACTUAL Y FUTURA

6.1 Estado Actual

En la actualidad el poder acceder a un producto o a un servicio desde nuestro domicilio y que nos llegue al mismo, resulta complejo y casi imposible en nuestro medio, ya que no contamos con dichos sistemas que presten las facilidades para hacer pedidos desde la comodidad y seguridad de nuestro hogar y menos a través de una página WEB. Por otro lado también consideramos cual es el estado de los actuales portales WEB, siendo estos de difícil acceso por su gran contenido multimedia haciendo que en el momento de cargar la página esta se torne lenta y a la vez consuma recursos de tiempo y dinero, considerando el estado actual de los proveedores de Internet los cuales cobran por evento llegando a costar 1 megabyte hasta 5.04 dólares en algunas compañías.

6.2 Estado Futuro

Lo que pretendemos conseguir con esta Tesis, es el desarrollo de un sistema de estregas y repartos que funcione en la WEB y que facilite a los usuarios el poder acceder en cualquier momento a un producto o servicio determinado. Otro resultado que buscamos obtener con este proyecto es que el software a desarrollar esté sustentado en las aplicaciones RIA (**Aplicaciones de Internet Enriquecidas**), siendo de gran utilidad acceder a esta aplicación aun cuando no se esté conectado al Internet pudiendo revisar los catálogos de productos o servicios y realizar pedidos con tan solo ejecutar la aplicación desde el escritorio y de esta forma reducir tiempos y costos a los usuarios.

7. MARCO CONCEPTUAL

7.1. Proyecto

Proyecto es un plan de trabajo, con acciones sistemáticas, es decir, coordinadas entre sí, valiéndose de los medios necesarios y posibles, en busca de objetivos específicos a alcanzar en un tiempo previsto.

Surge con una idea, para solucionar un problema, ya sea porque aún no se ha encontrado solución o porque las soluciones existentes no satisfacen las necesidades actuales.

7.1.1. Características de los Proyectos

Dentro de un proyecto, pueden distinguirse distintas etapas. En principio surge una **idea**, que establece la necesidad u oportunidad a partir de la cual se diseña el proyecto. Luego, en la etapa del **diseño** propiamente dicha, se realiza una valoración de las opciones y estrategias a seguir, con el objetivo a cumplir como guía. Seguido tenemos la **Prueba** que todo proceso una vez concluido tiene que ser sometido a diversas pruebas para verificar si cumple con lo establecido en sus requisitos.
Finalmente llega el momento de la **ejecución** y, una vez finalizada, se realiza la **evaluación** (cuando el proyecto es revisado y se juzgan sus resultados en relación a los objetivos planteados); **Duración,** Una vez el proyecto concluido, se lo debe mejorar, o innovar, de tal modo no quede obsoleto en el tiempo, y se ajuste a las necesidades actuales de los usuarios. **Riesgo** toda idea o proyecto a desarrollarse, tiene un alta tasa de riesgo de no satisfacer las necesidades como estuvieron planteadas inicialmente, o las que pueden ir surgiendo en el desarrollo del proyecto.

8. Marco Teórico

8.1 Aplicaciones de Internet Ricas

RIA, (Aplicaciones de Internet Enriquecidas) son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.
Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegado muchas veces, a recargar la misma página con un mínimo cambio.
Otra de las desventajas de las tradicionales aplicaciones Web es la poca capacidad multimedia que posee. En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce

comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos.
Hay muchas herramientas para la creación de entornos RIA. Entre estas se puede mencionar las plataformas Adobe Flash, Adobe Flex y Adobe AIR de Adobe.
Entre los beneficios principales de aplicaciones RIA tenemos un mejoramiento importante en la experiencia visual, que hacen del uso de la aplicación algo muy sencillo, ofrece mejoras en la conectividad y despliegue instantáneo de la aplicación, agilizando su acceso, garantizan la desvinculación de la capa de presentación es decir acceso a la aplicación desde cualquier computador en cualquier lugar del mundo.

8.2 Sistema Delivery (Entrega y Repartos)

Se trata de un sistema de hacer llegar productos o servicios hasta el consumidor. Donde las empresas son dueñas de fijar las expectativas en el mercado, pero una vez que lo hicieron, deben responder con un delivery adecuado.
La gestión del delivery se preocupa del diseño, planificación, implementación y mejoramiento de los flujos asociados a la entrega, generalmente sujeta a restricciones de tiempos y costos.

8.3 Comercio Electrónico

El comercio electrónico, también conocido como *e-commerce*, consiste en la compra y venta de productos o de servicios a través de medios electrónicos, tales como el Internet y otras redes de ordenadores.

8.4 Negocio al cliente (B2C - Business to Consumer):

En esta categoría se incluyen todos aquellos sitios de Internet que vendan cualquier tipo de producto al público en general, como por ejemplo, amazon.com. Entre algunas de las ventajas tenemos:

Grandes y pequeñas empresas pueden mostrar sus catálogos en línea y vender sus productos, sin incurrir en altas inversiones; comunicación directa con los clientes, sin intermediarios; no es necesario grandes cantidades de inventario físico para poder vender en línea, solo rápidas soluciones de distribución.

8.5 WEB 2.0

Es la revolución en la industria de la computación causada por el avance del Internet como plataforma, y el intento por entender las reglas para el éxito en una nueva plataforma (Oreilly).

8.5.1 Características

La Web 2.0 permite a los usuarios de la Internet la libertad para cooperar y ya no solo leer la información de las páginas. Esta evolución permite correr aplicaciones de software desde un servidor remoto de internet con el fin de que los usuarios tengan control sobre la información que ellos ingresan.
Esto ha permitido a los usuarios agruparse en torno a gustos, preferencias, y demás características que definen a la población, estos a la vez por medio de diversas aplicaciones pueden compartir fotos, documentos, videos, información personal, etc.

9. Herramientas a utilizar.

Las herramientas al igual que la tecnología a utilizar en el desarrollo de esta tesis, comprende tanto software como hardware que nos ayudará a llevar a cabo el cumplimiento de los objetivos antes planteados y en un tiempo determinado.

9.1 ADOBE FLEX

Flex es un marco de trabajo de código abierto gratuito altamente productivo para la creación y el mantenimiento de aplicaciones web expresivas que se implantan coherentemente en los principales exploradores, equipos de sobremesa y sistemas operativos. Mientras que las aplicaciones de Flex se pueden crear únicamente utilizando el kit de desarrollo de software de Flex*, los desarrolladores pueden utilizar el software Adobe® Flex® Builder™ 3 para acelerar la producción de páginas WEB.

9.2 ADOBE AIR

Es un entorno de ejecución multiplataforma para la construcción de aplicaciones RIA (*Rich Internet Applications*) utilizando Adobe Flash, Adobe Flex, HTML y AJAX, las cuales pueden usarse como aplicaciones de escritorio.
Adobe AIR es una tecnología que permite la creación de aplicaciones de escritorio (de propósito general) a partir de tecnologías de desarrollo de páginas web, como pueden ser HTML, Ajax o Flash. Con Adobe AIR los desarrolladores podemos aprovechar nuestros conocimientos web para hacer aplicaciones multimedia para el escritorio. Es

decir, a partir de un desarrollo de una aplicación web, crear una aplicación general con los mismos contenidos o utilidades del sitio web.

AIR intenta ser un entorno de ejecución versátil, ya que permite que el código Flash, HTML o JavaScript existentes sea reutilizado para construir programas tradicionales de escritorio. Sin embargo, en la mayoría de los casos, las aplicaciones ricas de Internet almacenan los datos de los usuarios en sus propios servidores, pero la capacidad para consumir y trabajar con datos en el sistema de archivo local de un usuario permite una mayor flexibilidad cuando una aplicación está trabajando sin conexión.

9.3 HTML (HyperText Markup Language)

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares.

9.4 PHP (Hypertext Preprocessor)

Es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

9.5 MySQL (Base de Datos)

Como base de datos consideramos hacer uso de MySQL debido a que es un sistema de gestión de base de datos relacional, multihilo y multiusuario. MySQL nos permitirá mantener los registros de la base de datos de los usuarios ordenadamente en tablas que se establecerán por medio de los análisis.

9.6 AJAX.

Acrónimo de ***A**synchronous **J**avaScript **A**nd **X**ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

9.7 Adobe Flash

Es una aplicación en forma de estudio de animación que trabaja sobre "Fotogramas" destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma.

9.8 ActionScript

ActionScript es un lenguaje de programación orientado a objetos (OOP), utilizado en especial en aplicaciones web animadas.
Es un lenguaje de script, esto quiere decir que no requiere la creación de un programa completo para que la aplicación alcance los objetivos. El lenguaje está basado en especificaciones de estándar de industria ECMA-262, un estándar para Javascript, de ahí que ActionScript se parezca tanto a Javascript.

10. **ESQUEMA TENTATIVO**

- Introducción
- Objetivos
    - General
    - Específicos

1. Investigación teórica.

1.1.1. *Nuevas tecnologías y herramientas para el desarrollo web (RIA*

1.2. *).*

1.2.1. Beneficios

1.2.2. Ejemplos RIA

1.3. *Características técnicas*

1.4. *Ventajas y Desventajas*

1.5. *Herramientas para el desarrollo*

1.6. *Costos Herramientas*

1.7. *Equipos para desarrollo*

2. Aprender el desarrollo de WEB con Herramientas RIA y tecnologías WEB

2.1. Adobe Air

2.1.1. SDK

2.2. Adobe Flex

2.2.1. Concepto y Fundamentos

2.2.2. Antecedentes

2.2.3. Inclusión de Herramientas

2.2.3.1. Java Script

2.2.3.2. ActionScript

2.2.3.3. Mxml

2.2.3.4. CSS Hojas de Estilo

2.2.3.5. Flash

3. Desarrollo de un Sistema para Comercio Electrónico de Entregas y repartos "Delivery"

3.1. Entrevistas a usuarios

3.2. Investigación del tema

3.3. Elaboración de requisitos

3.4. Análisis del Proyecto para la Aplicación Web

3.4.1. Requisitos para el análisis de las WebApps

3.4.2. Modelo del análisis para las WEB Apps

3.4.3. El modelo de contenido

3.4.4. El modelo de Interacción

3.4.5. El modelo funcional

3.4.6. El modelo de configuración

3.4.7. Análisis relación-navegación

4. Diseño para aplicaciones WEB

4.1. Introducción de diseño

4.2. Diseño de la interfaz de la WEB App

4.3. Diseño Estético

4.4. Diseño de Contenido

4.5. Diseño Arquitectónico

4.6. Diseño de Navegación

4.7. Diseño a nivel de Componentes

5. Implementación

5.1 Programación de la aplicación

5.1.1 Mantenimiento de usuarios

5.1.2 Mantenimiento de proveedores

5.1.2 Pedidos

5.1.2.1 Recepción de pedidos

5.1.2.2 Entrega de pedidos

5.2 Adaptación de la aplicación web a una aplicación de escritorio

5.2.1 Programación del ejecutable de la aplicación con Adobe Air

6. Pruebas de la Aplicación Web

6.1. Estrategias de pruebas
6.2. Prueba de Contenido
6.3. Prueba de Interfaz del Usuario
6.4. Prueba al nivel de Componentes
6.5. Pruebas de Navegación
6.6. Pruebas de Configuración
6.7. Pruebas de Seguridad
6.8. Pruebas de Desempeño.

7. Conclusiones y Documentación

11. PROCEDIMIENTOS METODOLÓGICOS

Para la investigación de este proyecto nos basaremos en la siguiente bibliografía:

Información de Tecnologías RIA.

http://www.informatica-hoy.com.ar/redes/RIA-es-el-futuro-de-la-navegacion-en-

http://www.usolab.com/articulos/desafios_interfaz_web_2.php

Información de Adobe Flex

http://www.adobe.com/es/products/flex/

Información de Adobe Air

http://www.adobe.com/es/products/air/business/

Información de Delivery

http://es.wikipedia.org/wiki/Delivery

Información de Comercio electrónico.

http://www.panamacom.com/que_es_ec.html

http://www.degerencia.com/tema/comercio_electronico

Andrew Grove, Estrategia De Negocios Electrónicos,

http://sabanet.unisabana.edu.co/comercio/conceptos_comercio_electr%C3%B3nico.htm

Información de MYSQL gestor de Base de Datos

http://www.mysql.com/

Información de PHP

http://www.php.net/

Información sobre el servidor Web Apache APPSERV

http://www.apache.org/

http://www.appserv.org

Regulación y estándares para la creación de Páginas WEB.

http://www.w3.org/

12. **RECURSOS HUMANOS Y TÉCNICOS**

Recursos Humanos

Los temas propuestos en esta tesis serán investigados y desarrollados por:

- Daysi Marlene Becerra Naranjo.
- Juan Andrés Vélez Zea.

Director de Tesis:

Ing. Pablo Pintado

Recursos Materiales y Técnicos:

- Notebook AMD Turion 64x2 1.8 ghz; 3 Gb de RAM. SO. Windos XP SP3 32 bits.
- Notebook AMD Turion 64x2 2.1 ghz; 4 Gb de RAM. SO. Windows Vista Ultimate 64 bits.

Cronograma Tentativo de Actividades:

Actividades / Fecha	MES 1				MES 2				MES 3				MES 4				MES 5				MES 6				MES 7				MES 8				MES 9			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2		
1. Investigación teórica																																				
2. Aprender uso de las herramientas																																				
3. Desarrollo de un Sistema para Comercio Electrónico de Entregas y repartos "Delivery"																																				
4. Diseño de la aplicación WEB																																				
5. Implementación																																				
6. Pruebas de la Aplicación																																				
7. Gestión de riesgos																																				
8. Conclusiones y Documentación																																				