UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO-CHILE

"SISTEMA PARA LA IMPLEMENTACIÓN MASIVA DE DELIVERY ONLINE DE COMIDA"

KLAUS DANIEL HOTT VIDAL
SEBASTIÁN ANDRÉS TORO OYARZÚN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESORES GUIA: WERNER CREIXELL F.
AGUSTIN GONZALEZ V.

ABRIL - 2011

SISTEMA PARA LA IMPLEMENTACIÓN MASIVA DE DELIVERY ONLINE DE COMIDA

Memoria para optar al título de Ingeniero Civil Telemático
KLAUS DANIEL HOTT VIDAL
SEBASTIÁN ANDRÉS TORO OYARZÚN
Profesores Guías: Werner Creixell F, Agustín González V.
Abril 2011

Resumen

El presente informe detalla los requerimientos, el diseño, la implementación y testing de un sistema, que permite a cualquier restaurante implementar su propio servicio de reparto a domicilio online. Esto es, cada restaurante facilitará una interfaz web para sus clientes y así, estos puedan realizar pedidos en Internet. El proyecto se dividió en tres grandes módulos: un módulo administrador, en la cual la empresa dirigirá su negocio de pedidos online; una interfaz web para el cliente, en la que los consumidores del restaurante podrán ingresar e informarse acerca de los productos que oferta el mismo, para luego gestionar un pedido si así lo desean; y un servidor, encargado de alojar a todos los restaurantes y de proporcionarles las herramientas y servicios necesarios para su correcto funcionamiento. Los primeros dos módulos fueron realizadas usando el framework de Flex, debido a su especialización para desarrollar aplicaciones de internet ricas en contenido (RIAs). El módulo del servidor fue gestionado usando el “Zend Framework”, enfocado al desarrollo de aplicaciones y servicios web, por cuanto agiliza el trabajo y ayuda a la potencial escalabilidad del sistema.

Tras concluir la construcción del presente trabajo, se obtuvo un sistema absolutamente modular, cuya funcionalidad será fácilmente extensible. Sin embargo, no se logró implementar un sistema que fuese suficientemente escalable como para abarcar el mercado a nivel nacional, sino sólo a nivel regional. A continuación se detallan cada uno de los procesos de creación de esta herramienta gastronómica y las conclusiones acerca de su desempeño.

Palabras clave: Sistema de implementación masiva, pedidos a domicilio online.

MASIVE IMPLENTATION OF ONLINE FOOD DELIVERY SERVICES

Engineering Project to obtain the degree of Ingeniero Civil Telemático

KLAUS DANIEL HOTT VIDAL

SEBASTIÁN ANDRÉS TORO OYARZÚN

Advising Professors: Werner Creixell F, Agustín González V.

April 2011

Abstract

This report details the requirements, design, implementation and testing of a system, which allows any restaurant to set up their own online delivery service. In other words, each restaurant will provide a web interface to their clients, so that they can place orders over the Internet. The project was divided into three modules: an administrator module, which will allow the restaurant to manage their online business; a client interface, where the clients of the restaurant will be able to look at the products and then make an order if they want to; and a server, in charge of storing the restaurants and providing them the necessary set of tools and services. The first two modules were developed using the Flex Framework, because of its specialization in creating rich-content internet applications (RIAs). The server was built using the Zend Framework, specialized in the development of web applications and services, because it speeds up the development process and it helps with the scalability of the system.

After the project was finished, the result was a completely modular system, which its functionality will be easy to increase. However, the desired scalability was not reached. The system will not be implemented on a national level, but only on a regional level. Every process that was part of this gastronomical tool and the conclusions relative to its performance are detailed next.

Keywords: Massive implementation system, online food delivery.

Glosario

- ACID: “Atomicity, Consistency, Isolation and Durability”.
- AJAX: “Asynchronous JavaScript And XML”.
- ALT: “Alternative”.
- AMF: “Action Message Format”.
- AMFPHP: Implementación de AMF en PHP.
- AS: “Action Script”.
- CPU: “Central Prossesing Unit”.
- CRUD: “Create, Read, Update and Delete”.
- DIV: “Divition”.
- GPS: “Global Positioning System”.
- HTML: “HyperText Markup Language”.
- IDE: “Integrated development environment”.
- IMG: “Image”.
- JSON: “JavaScript Object Notation”.
- KB: “Kilo Byte”.
- Mb: “Mega Bit”.
- MB: “Mega Byte”.
- MVC: “Model View Controller”.
- MXML: “Macromedia Extensible Markup Language”.
- PHP: “Hypertext Preprocessor”.
- PHTML: HTML con algo de código PHP.
- PyME: Pequeñas y Mediana Empresa.
- RAM: “Random Access Memory”.
- RIA: “Rich Internet Application”.
- SEO: “Search Engine Optimization”.
- SIMDOC: “Sistema para la Implementación Masiva de Delivery Online de Comidas”.

- SOA: “Service Oriented Architecture”.
- SWF: “ShockWave Flash” o “Small Web Format”.
- URL: “Uniform Resource Locator”.
- XML: “Extensible Markup Language”.
- WYSIWYG: “What You See Is What You Get”
- WWW: “World Wide Web”.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

Hoy en día, las telecomunicaciones han logrado penetrar absolutamente en todas las áreas de la sociedad, tanto en el diario vivir de una familia como en la manera de realizar negocios de una empresa. Desde hace algunas décadas, las telecomunicaciones han sido utilizadas para apoyar y colaborar en la gestión de los negocios; el fax y el teléfono fijo fueron y son, herramientas que ayudaron a disminuir increíblemente los tiempos de todos los procesos relacionados con la comunicación entre diferentes entidades comerciales. Luego con la aparición del teléfono celular, muchas empresas se vieron inmensamente beneficiadas al poder contactarse con sus trabajadores que estaban fuera de las oficinas y así poder entregarles las directrices necesarias para sus labores. Más tarde, con la presentación del GPS, las compañías dedicadas al transporte o al reparto de productos a domicilio, se vieron favorecidas con esta herramienta al poder controlar el comportamiento de su flota y perfeccionar la planificación de sus rutas, concluyendo en una optimización del uso de sus recursos para su negocio. Se ha visto como las tecnologías creadas en el área de las telecomunicaciones han tenido una directa repercusión en las empresas comerciales, ya sea en sus modelos de negocios, logística, producción, etc. Es por esto, que se ha decidido desarrollar un sistema que utilice las actuales tecnologías de telecomunicaciones para generar beneficios para las empresas.

Si bien son muchas las tecnologías existentes en esta área, ha llamado fuertemente la atención la plataforma de Internet para ser usada como la base de este proyecto. Una de las razones por las que se eligió esta plataforma, es su alta tasa de penetración y de crecimiento. A continuación, se detallan estadísticas obtenidas por Miniwatts Marketing Group, referentes al uso de internet a nivel mundial:

Tabla 1-1: Estadísticas de Población y de Uso de Internet a Nivel Mundial [1]

Región Mundo	Población (2010 Est.)	Usuarios 31/12/2000	Usuarios Junio 2010	Penetrac (% Pob)	Crecim. 2000-2010	% Usuarios
África	1,013,779,050	4,514,400	110,931,700	10.9 %	2,357.3 %	5.6 %
Asia	3,834,792,852	114,304,000	825,094,396	21.5 %	621.8 %	42.0 %
Europa	813,319,511	105,096,093	475,069,448	58.4 %	352.0 %	24.2 %
Medio Or	212,336,924	3,284,800	63,240,946	29.8 %	1,825.3 %	3.2 %
Am del N	344,124,450	108,096,800	266,224,500	77.4 %	146.3 %	13.5 %
Am L y C	592,556,972	18,068,919	204,689,836	34.5 %	1,032.8 %	10.4 %
Oceania	34,700,201	7,620,480	21,263,990	61.3 %	179.0 %	1.1 %
Total	6,845,609,960	360,985,492	1,966,514,816	28.7 %	444.8 %	100.0 %

La tabla 1-1 muestra la alta penetración de usuarios a nivel mundial, lo cual se traduce en una demostración de la trascendencia que ha cobrado Internet en la vida cotidiana de las personas (ergo las altísimas tasas de crecimiento). En las regiones desarrolladas, destaca el hecho que las tasas de crecimiento no son tan altas, debido a que su porcentaje de penetración es muy elevado; en cambio, en las regiones aún no desarrolladas, las tasas de crecimiento son desmesuradas. Es posible concluir que una de las razones a las que responde este comportamiento se basa en que la WWW es una plataforma base para el desarrollo de innumerables empresas y es usada en estos casos para aumentar las ventas de muchos negocios ya establecidos. Hoy, es de radical importancia para una compañía tener salida a Internet, ya que sus clientes, proveedores y sus pares lo utilizan para comunicarse. Una entidad comercial que no hace un buen uso de las tecnologías informáticas y comunicacionales, sin importar la magnífica estrategia de ventas y de marketing que pudiera tener, está desaprovechando un medio de comunicación que cada día cobra mayor relevancia. Es por esto, que se hace imperativo el desarrollo de aplicaciones y herramientas para las entidades comerciales, con el fin de diversificar y optimizar sus negocios.

1.1 Identificación del problema

En la actualidad, las grandes empresas invierten en herramientas de software tanto para la gestión de sus negocios como para diversificar sus modelos de negocios. Sin embargo, esto sucede en una menor medida en el caso de las PyMEs. Según un estudio realizado por Pyramid Research [2], la adopción de herramientas online para ventas y marketing es aún incipiente en la región pero con un potencial de crecimiento muy acelerado. De esta investigación se obtuvo que sólo el 18% de los sitios web perteneciente a PyMEs, permite realizar transacciones web. La indagación arrojó que cada vez es mayor el número de PyMEs interesadas en lanzarse al mercado web, por lo tanto es sólo una cuestión de tiempo que el porcentaje más importante de las PyMEs cuente con su propia página web para promoverse.

Complementar un negocio con una plataforma online, requiere de un desembolso que es difícilmente abordable para ciertas empresas. Como las sociedades comerciales están constantemente realizando egresos por diversos motivos, algunas de ellas no están en condiciones o no desean efectuar una inversión adicional, aunque ésta les vaya a brindar interesantes beneficios y estén conscientes de ello.

A continuación se detallan los gastos mínimos necesarios para implementar un servicio de ventas online:

- Se debe adquirir los servicios de un servidor de webhosting, que albergue el sitio a desarrollar. Según los presupuestos obtenidos en el estudio que se encuentra en el Anexo A: Presupuesto Para un Sitio Web, las tarifas oscilan alrededor de $23.000.
- Cuando uno contrata un servicio de Webhosting, debe también adquirir un dominio para asociar a su cuenta. En este caso se consideró un dominio “.cl” cuyo costo es de $19.000 [3].
- Por último, lo más importante es el diseño del sitio web y la implementación de la vitrina de productos junto con las transacciones web. Según las cotizaciones recibidas, el promedio de este servicio varía en valores cercanos a $400.000 (ver Anexo A: Presupuesto Para un Sitio Web)

El primer y tercer punto fue evaluado solicitando cotizaciones a diversas empresas que se dedican a entregar dichos servicios. Los dominios web ".cl" están a cargo sólo de la Universidad de Chile, por lo tanto, las tarifas son únicas. Luego de analizar los costos mencionados anteriormente, se llega a un mínimo de desembolso de $440.000 (sin IVA). Para muchas empresas de mediana y pequeña envergadura no es un monto fácil de desembolsar, sobre todo para estas últimas.

Debido al auge que ha sufrido Internet, es de vital importancia que las compañías vendan o promuevan sus productos por este medio. Esta es una manera rápida, cómoda y eficiente en la cual los clientes puedan ponerse al tanto de cuáles son los productos ofrecidos por las empresas y obtener información detallada acerca de ellos. Es por esto, que se ha decidido centrar la atención en generar una herramienta que ayude a las empresas a ampliar su margen de ventas utilizando el medio de Internet, de una manera económica y abordable.

Específicamente, se ha determinado dirigir este proyecto hacia el área gastronómica, es decir, empresas que se dediquen a la venta de alimentos, ya sean restaurantes, minimarkets, servicios de catering, etc. Ahora en adelante, cuando se empleen los términos "empresas", "negocios", etc. de forma general, se estará haciendo referencia a este subtipo de empresas. Éstas también presentan el problema mencionado anteriormente, sin embargo, la inmensa mayoría de ellas tampoco cuenta con un servicio de reparto a domicilio. Este último punto, implica desaprovechar una oportunidad de negocio importante para dichas empresas.

Muchas de estas entidades gastronómicas se enfrentan a distintos problemas en el día a día, como por ejemplo:

- Durante los fines de semana, horas de almuerzo, etc., muchos restaurantes están copados. No cuentan con mesas para atender a más clientes, y por lo mismo, pese a que algunos cuentan con capacidad en la cocina, pierden esa oportunidad debido a su falta de espacio físico.

- Para la inmensa mayoría de empresas gastronómicas, es necesario realizar gastos para el arriendo o adquisición de un local. Si las empresas implementasen sus negocios por Internet, este gasto dejaría de ser indispensable.
- Este tipo de sociedades comerciales están sometidos a variaciones en su demanda, y por lo mismo se ven afligidos por este problema. Un sistema que permita la venta de sus productos en Internet, aumentaría potencialmente su demanda.

A lo anterior se debe agregar, que ciertas empresas gustarían de un servicio que se encargue de llevar el almuerzo o colación para sus empleados al lugar de trabajo, ya sea para el día a día o para cuando haya reuniones con clientes.

Son diversos los motivos por los cuales una empresa del área gastronómica debiese implementar un sistema de reparto a domicilio, es por ello, que se ha considerado abarcar también esta problemática en el presente trabajo.

Estas empresas gastronómicas, necesitan resultados tanto para el asunto del posicionamiento en Internet, como para el del reparto a domicilio. Cuanto mejor si ambos son resueltos con una misma solución, que es lo que se perseguirá con este proyecto.

1.2 Metodologías usadas actualmente

Existen distintas formas para enfrentar el problema de difundir un servicio de reparto a domicilio, de ahora en adelante delivery, entre los posibles clientes:

- Los actuales clientes de la empresa la recomiendan a otros potenciales futuros clientes. En la mayoría de estos casos el contacto entre cliente y empresa es realizado por teléfono o celular. En el último tiempo, también se ha aprovechado la promoción de servicios a través de redes sociales, donde el “modus operandi” del negocio se repite, sólo que esta vez es a través del e-mail o mensajes a través de estas redes.
- Las entidades comerciales que cuentan con los medios y construyen una página web, por lo general, se quedan en la promoción del local y a lo sumo una breve

descripción de la carta. Esta falta de información obliga al cliente a hacer uso de la información de contacto, expuesta en la página.

- Las páginas amarillas son el siguiente paso para aquellas empresas dispuestas a pagar, logrando promocionarse a través del papel y la web, entregando sólo la información de contacto necesaria.
- Últimamente han nacido empresas cuyo único propósito es encargarse de la logística del reparto, logrando que los restaurantes se preocupen sólo de la elaboración de la comida. Estas empresas hacen de nexo, entre el cliente y el restaurant y operan tanto vía web como telefónica. Pese a que este ítem pertenece a otra categoría y no a “medios para difundir el servicio”, se tomó en cuenta debido a que de igual manera dichas entidades gastronómicas están promoviendo sus platos y sus menús, a través de la empresa encargada de llevar la comida.

1.3 Herramientas existentes

Para apoyar las metodologías expuestas en la sección anterior se hará referencia a herramientas que, de alguna manera intentan cubrir la necesidad.

- Software de Restaurant: Algunos restaurants, administran su negocio a través de un software que comunica la caja con la cocina, para evitar entradas y salidas innecesarias. Este tipo de software también ha sido usado para asociar los pedidos a una dirección, en vez de una mesa en particular, de este modo el restaurant obtiene un registro de los pedidos solicitados; sin embargo es imperioso que exista un telefonista que ingrese los pedidos al sistema. Ejemplos de estos software son Innoves, Methodo, Resto y Laudus. [4][5][6][7]
- Carros de Compra Online: En el caso de negocios masivos y con una gran área de cobertura, se hacen imprescindibles. Sin embargo, por parte de los empresarios PyMEs, es considerado poco útil para realizar ventas locales, desaprovechando así, parte de los posibles compradores. Este tipo de herramienta está diseñado para tiendas y para aplicarlo al área de los restaurants se requeriría necesariamente de una capacitación para el personal.

1.4 Objetivos del Proyecto

En base a los problemas planteados en esta sección, se propone como objetivo el desarrollar un sistema que permita la implementación de un servicio de reparto a domicilio para múltiples empresas gastronómicas a nivel nacional. Este sistema deberá posicionar a las entidades comerciales en Internet y hacerlas visibles al público. Les facilitará las herramientas necesarias para administrar su sistema de delivery además de una página web en la cual ellos podrán promover sus productos y ofrecer la venta online de estos. Todo lo anterior debe lograrse de tal manera, que permita la adquisición del servicio, por parte de las PyMEs, de manera económica, es decir, que sea accesible su implementación por parte de estas empresas.

2 Sistema para la Implementación Masiva de Delivery de Comida Online

En este capítulo se buscará fijar con claridad y exactitud cuál será el producto a desarrollar. Se definirán los beneficios que entregará a sus usuarios y los requerimientos necesarios para poder cumplirlos. Desde ahora en adelante se le llamará SIMDOC al sistema a implementar.

2.1 Modelo de negocios del sistema

En esta sección se buscará definir todas las aristas relacionadas con el modelo de negocios del producto, esto incluye su delimitación, determinar quiénes serán los clientes que contraten el servicio y quiénes serán sus usuarios, etc. Además se explicará cómo éste, logrará diferenciarse de las aplicaciones mencionadas en el capítulo anterior.

2.1.1 Definición del producto

El sistema permitirá que múltiples empresas implementen su propio servicio de reparto a domicilio. Cada una de ellas podrá registrar sus productos en el sistema, los cuales podrá ofertar al público consumidor y así este último podrá realizar sus pedidos directamente en el sitio de la empresa en cuestión.

A los establecimientos se les ofrecerá un software que les permitirá implementar íntegramente su propio negocio de delivery. Es decir, contempla todas las herramientas necesarias para una completa gestión de un sistema de reparto a domicilio. Se debe destacar, que si bien la definición recién formalizada puede abarcar cualquier empresa que desee ofrecer sus productos de manera “online” para luego repartirlos a domicilio, se ha decidido acotar la amplia gama de empresas a sólo las dedicadas al área gastronómica. Esto se debe, a que existen algunas diferencias en cuanto a la implementación final del sistema y por razones de tiempo, se prefirió ceñirse a este caso particular.

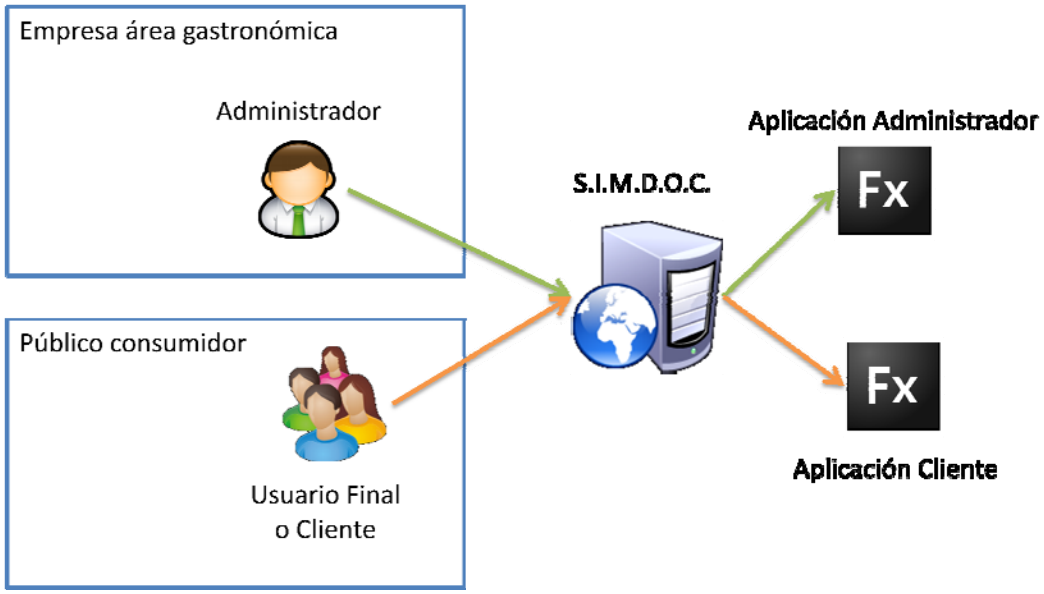

Figura 2-1: Diagrama explicativo de los actores del sistema

En la figura 2-1 se definen los actores que formarán parte del sistema. Para poder entender más fácilmente las explicaciones que siguen más adelante, se debe explicar quién es cada actor. Se le llamará **usuario final** o **cliente,** al público que consumirá los productos entregados por las empresas gastronómicas (desde ahora empresas), quien hará uso de una aplicación llamada **aplicación cliente**. **Administrador,** será el usuario al que la empresa le encargue utilizar las aplicaciones que permitirán la implementación del sistema de delivery, Este usuario hará uso de una aplicación llamada **aplicación administrador**.

El sistema ofrecerá a cada empresa tres ítems: una aplicación de control y gestión para el negocio de delivery (aplicación administrador), una aplicación que contará con una vitrina para ofrecer los productos y un carro de compras para poder realizar los pedidos (aplicación cliente). Por último un set de instrucciones detalladas que le permitirán al administrador implementar la aplicación cliente.

La aplicación administrador consta de seis módulos:

- Uno encargado de administrar los productos que posee la empresa.
- Otro que tiene como función administrar los repartidores de la empresa.
- Uno que se ocupa únicamente de gestionar los pedidos que realizan los clientes.

- Un módulo responsable de realizar pedidos o modificar los que el administrador desee.
- Un módulo de estadísticas, destinado a mostrar gráficos relacionados con el área de delivery del negocio.
- Un módulo encargado de definir ciertos aspectos del negocio: como el área de cobertura, las contraseñas de acceso, los cobros por delivery, etc.

La aplicación cliente, es la que permite al usuario final observar, elegir y luego comprar los productos disponibles que tiene la empresa para la venta online. Tanto los productos, como la apariencia del software serán editables por la aplicación administrador, esto con el fin de entregarles a sus clientes la imagen corporativa de la empresa en cuestión.

A la empresa no se le entregará ningún software sino que ambas aplicaciones se ejecutarán desde el servidor. Sólo se le entregará instrucciones detalladas con las cuales podrá mostrar (referenciar) la aplicación cliente en cualquier blog o página web. En caso de no ser de su interés, podrá entregar a **sus clientes** la dirección de su aplicación cliente en el servidor de sistema.

2.1.2 El cliente de SIMDOC

Este sistema se enfoca a pequeñas y medianas empresas, en el área de la gastronomía, más específicamente a la venta de los productos cocinados. Éstos, son los considerados como clientes por SIMDOC y puede ser cualquiera de los siguientes tipos:

- De Pastelería y Repostería.
- De Coctelería y Banquetería
- Restaurantes que no poseen servicio de “delivery” implementado.
- Restaurantes con servicio de “delivery” implementado pero sin su propio carro de compras online.

El proyecto fue ideado para satisfacer las necesidades de las empresas recién mencionadas, pero se puede ampliar la oferta para satisfacer las necesidades de otros tipos de negocios, sólo deben cumplir con los siguientes requerimientos:

- Desear realizar ventas de sus productos online.
- Que sus productos sean distribuidos y entregados al domicilio de su cliente (usuario final).
- Que el cobro deba efectuarse contra entrega.

A continuación se exponen algunos ejemplos:

- Panaderías
- Minimarkets
- Verdulerías
- Carnicerías, etc.

Las empresas definirán a qué mercado enfocarán sus negocios y por ende, ellos determinan quién es el usuario final de su aplicación cliente, los únicos requerimientos con los que estos usuarios deben cumplir son:

- Desear ahorrar tiempo en la compra.
- Ser partidario y propenso a realizar compras por internet (en este caso no se paga por internet, lo cual es una ventaja al momento de comprar).

2.1.3 ¿Cómo diferencia el producto?

Esta herramienta posee tres ventajas competitivas con respecto a sus símiles, explicadas en la sección 1-3, que son:

Servidor: Las empresas necesitan contratar servicios para la creación, implementación y administración de su página web. Además de lo anterior, deben adquirir un servicio de "web hosting" para su sitio. Por lo tanto, se deduce que son múltiples los gastos necesarios para que una empresa pueda ofrecer de manera online sus servicios. SIMDOC permite al cliente presentar sus productos online sin necesidad de adquirir servicios externos, ya que puede ser implementado en un "Blog" o en "Facebook". Ambas plataformas sociales son conocidas y usadas por la inmensa mayoría de los usuarios de internet. El hecho que esta herramienta pueda implementarse en esas plataformas, entrega una ventaja competitiva que no es menor en relación a sus competidores. Los interesados

no tendrán que incurrir en gastos adicionales al pago del servicio a menos que así lo deseen, ni deberán poseer conocimientos avanzados en el área o pagar a una empresa para que administre su sitio o para ponerlo en funcionamiento.

“Bugs” y versiones: Cada herramienta desarrollada posee errores en el código, los expertos en programación y en el desarrollo de software han establecido métodos de trabajo para minimizar la cantidad de bugs en un software. Sin embargo, la conclusión sigue siendo la misma: el software siempre traerá consigo errores. Este servicio se ofrecerá en forma de “player”, es decir, existirá una única aplicación administrador y una única aplicación cliente para todos los restaurantes. El hecho que todos los usuarios trabajen con la misma versión del programa, permitirá encontrar los “bugs” con mayor rapidez. Cabe señalar, que el trabajar con una única aplicación para todos los usuarios permite alcanzar altos niveles de calidad, confiabilidad y robustez del software; lo anterior se debe a que los ofertantes del servicio deberán mantenerlo en óptimas condiciones ya que un problema que aqueje a una empresa, afectará también a todas las restantes. Además se debe destacar que cada cambio, mejora u arreglo que se le introduzca al sistema, será propagado instantáneamente hacia todos los “players”, entregando la última versión del programa a todos los clientes.

Mantención no invasiva: Generalmente el ítem “mantenciones del software” es un inconveniente tanto para quién desarrolla el software como para quién lo utiliza, ya que ambos se tienen que adecuar a la agenda de su contraparte. A los clientes generalmente les incomoda el hecho que tengan que aceptar y recibir a una persona en su oficina para solucionar los problemas relacionados con sus equipos, también les es molesto el tiempo de espera que le toma al técnico en llegar a solucionar los problemas. Ante esta problemática, se pensó en un sistema en el que todas las mantenciones sean realizadas de forma “online”, eliminando las incomodidades, es así que estas mantenciones serán realizadas en horarios de baja demanda, probablemente de madrugada, con el motivo de no afectar a los clientes que contraten el sistema.

2.1.4 Utilidades para las empresas

El sistema está enfocado a múltiples tipos de clientes, por lo tanto, se puede hablar también de múltiples tipos de utilidades, las que serán detalladas a continuación.

Los negocios que poseen reparto a domicilio pero que lo realizan mediante llamadas telefónicas, podrán contar con una interfaz web que les permitirá a sus clientes acceder a la información de manera mucho más rápida y cómoda. Un número reducido de empresas con servicio de “delivery” tienen página web, que generalmente contiene sólo un catálogo muy limitado. SIMDOC les servirá también como un sistema de difusión para todos sus productos, cuya información se podrá modificar de manera sencilla y sin necesidad de contratar servicios de administración de su sitio.

A las empresas que no ofrecen comida a domicilio, esta herramienta les ayudará a percibir ganancias que anteriormente no recibían, ya que podrán poner en funcionamiento un sistema de “delivery” y así ampliar su margen de ventas.

Por otro lado, el sistema ofrece una herramienta de administración para operar el sistema de “delivery” de manera segura. Se mantendrá un registro de los usuarios finales que han realizado pedidos al restaurante, con el fin de identificar cuáles han intentado engañar al sistema y cuáles son fiables. Además cabe mencionar que el software de administración también es una herramienta para el control de gestión y de calidad del servicio prestado por el restaurante, ya que posee un módulo de estadísticas cuyo fin es controlar el desempeño de los repartidores, analizar la popularidad de los platos y las opciones más solicitadas.

Con todos los beneficios mencionados, el sistema asegura la entrega de una herramienta integral para la implementación del servicio de “deliveries”, o para la consolidación de aquellas empresas que ya lo poseen.

2.1.5 Beneficio económico

Hoy en día, existen múltiples formas de obtener un beneficio económico al momento de implementar el sistema en cuestión. A continuación, se detallan los más convenientes para SIMDOC, según lo planteado por diferentes autores: [8][9][10]

- **Servicio masificado:** Se trata de vender el producto, en forma de servicio, a la mayor cantidad posible de pequeñas y medianas empresas. Si se vendiese el software, su valor sería mayor a los $500.000 pesos, por lo tanto, por su alto valor

muchas empresas no estarían dispuestas a comprarlo. Sin embargo, al ofrecerse en forma de servicio, con un costo asequible mensual por su uso, un mayor número de empresas estarían interesadas en adquirirlo. Esto se debe a que la potencial ganancia mensual que obtendrían del servicio, sería mayor al monto que estarían costeando por el mismo.

- **Ganancias por transacciones:** Como el sistema está enfocado en generar más negocio para sus clientes, SIMDOC puede percibir una pequeña fracción del total de cada compra realizada a cada empresa. Obteniendo ganancias para el sistema por el alto número de transacciones mensuales que se generarán entre todas de las compañías.
- **Publicidad:** Tanto la aplicación administrador como la aplicación cliente contarán con "banners", los que serían utilizados por empresas externas al sistema para generar publicidad dinámica. Este tipo de publicidad no guardaría ningún tipo de relación con la empresa que hace uso de SIMDOC. El sistema efectuaría una cobranza a dichas empresas encargadas de facilitar la publicidad encontrada en los banners. Un ejemplo de este tipo de empresas es TradeDoubler (http://www.tradedoubler.com).
- **Servicio "Premium":** Este es una extensión de la categoría servicio masificado. Si bien existe un cobro por un servicio base, el sistema puede seguir desarrollándose y generando nuevas herramientas para la gestión del negocio de delivery de cada empresa. Se cobrará por un servicio Premium, en el cual se le entregará al cliente de SIMDOC módulos adicionales de los que cuenta el servicio básico. Por ejemplo: un módulo encargado del estudio de mercado.

La tercera alternativa (publicidad) llamó fuertemente la atención, ya que se puede complementar con cualquiera de las otras alternativas; sin embargo, a algunas empresas no les agradará contar con publicidad ajena en su página. La última opción es también interesante pero debido al limitado número de módulos que se desarrollarán inicialmente, no se puede implementar esta estrategia.

Tras estudiar las restantes posibilidades, se decidió optar por la alternativa "ganancias por transacciones". Esto se debe a que algunas de las empresas que se asocien al

sistema, estarán recién implementando su nuevo servicio de reparto a domicilio. La inversión que realicen éstas, irá asociada del éxito que tengan en la implementación de su negocio, por lo tanto, se les cobrará sólo un porcentaje de lo que hayan ganado con su nuevo servicio de reparto. En otras palabras, si el negocio no tuvo éxito el cobro será mínimo, sin embargo, si es exitoso el cobro será mayor. Lo mismo aplica para las empresas que ya cuentan con un servicio de delivery, a las que perciban ganancias a través de sus aplicaciones cliente se les cobrará por cada una de sus transacciones. La alternativa "servicio masificado" no cuenta con los atractivos ya mencionados, es por ello que se decidió optar por "ganancias por transacciones".

2.2 Análisis y definición de requerimientos

En esta sección, se describirán los requerimientos necesarios para que el sistema a desarrollar se diferencie de sus competidores y logre entregarles a sus clientes, las utilidades prometidas. Estos requisitos se pueden definir en dos categorías: las exigencias funcionales y las no funcionales. Los funcionales son aquellos perceptibles por el administrador y el usuario final y que son directamente relevantes para ellos; en cambio los no funcionales son los que sólo son atingentes al sistema, ayudando a disminuir los costos de implementación y operación del mismo.

2.2.1 Requerimientos funcionales

- La aplicación administrador debe contar con todos los módulos mencionados en la sección 2.1.1. Todos son necesarios para operar el negocio de reparto a domicilio.
- La aplicación cliente debe ser incrustable en "Facebook", en "Blogs" y en "Html".
- La fachada debe ser personalizable, para entregar la imagen corporativa del negocio.
- Desarrollar un sistema de compra, donde el cliente no tenga la necesidad de registrarse en el sistema pero sin que esto afecte la seguridad de la compra.

2.2.2 Requerimientos no funcionales

- Diseñar un sistema que sea capaz de manejar un alto número de negocios, a nivel nacional.
- Debido a que el sistema tiene como objetivo una implementación a nivel nacional, éste contará con un alto número de negocios inscritos, por lo tanto, el servidor estará contantemente sometido a una alta carga. Es necesario que la aplicación cliente sea diseñado para aminorar la carga del servidor, realizando la mayor cantidad de procesamiento posible, enviando al servidor sólo lo que es indispensable.
- Proveer una plataforma segura, con el fin de que sólo la empresa en cuestión pueda administrar su negocio.
- Tanto la aplicación cliente como la administrador deben diseñarse de manera de que sus usuarios sean capaces de utilizarlas sin necesidad de instrucciones ni ayuda externa, por lo tanto deben ser altamente intuitivas y de muy fácil manejo.
- Implementar una solución, en la cual los clientes y los usuarios finales trabajen siempre con la última versión del software, donde los cambios realizados a las aplicaciones se propaguen instantáneamente hacia todos los negocios.
- El sistema debe ser capaz de exponer la información que se mostrará en el “Flash Player”, de modo que ésta pueda ser indexada por distintos motores de búsqueda. Además, esto ayudará a que usuarios que no posean el software requerido, puedan ver la aplicación completa.

3 Diseño de la aplicación

Teniendo en cuenta los requerimientos y previo al desarrollo de cualquier solución, es necesario hacer un análisis al problema y definir distintos elementos que ayudarán organizar de forma más ordenada el software a construir. Los diagramas y arquitectura definidos en este capítulo, serán los planos para el futuro sistema.

3.1 Descripción General del Sistema

El sistema completo considera dos usuarios: el usuario final y el administrador. Y para cada uno de estos crear una aplicación acorde a las tareas que realizará cada uno, y los permisos que tenga.

Para el usuario final es necesaria una aplicación cliente que muestre las categorías y platos del restaurante, que permita al usuario agregar dichos platos a un carro de compras y efectuar un pedido al restaurante, con los ítems seleccionados.

La aplicación del administrador del restaurante requiere mayor trabajo ya que no sólo abarca el agregar, modificar y borrar platos del restaurante. Además se encuentran involucrados la administración de repartidores, modificación de pedidos, estadísticas y modelo de negocio del reparto.

En la sección 3.3 se estudiará con mayor detención, los casos de uso del sistema.

3.2 Casos de uso Aplicación Cliente

Como se ve en la figura 3-1, la aplicación cliente tiene 3 casos de uso y 2 actores: el usuario final y el motor de búsqueda. Este último corresponde a cualquier “web crawler” que revise la aplicación para indexar su contenido.

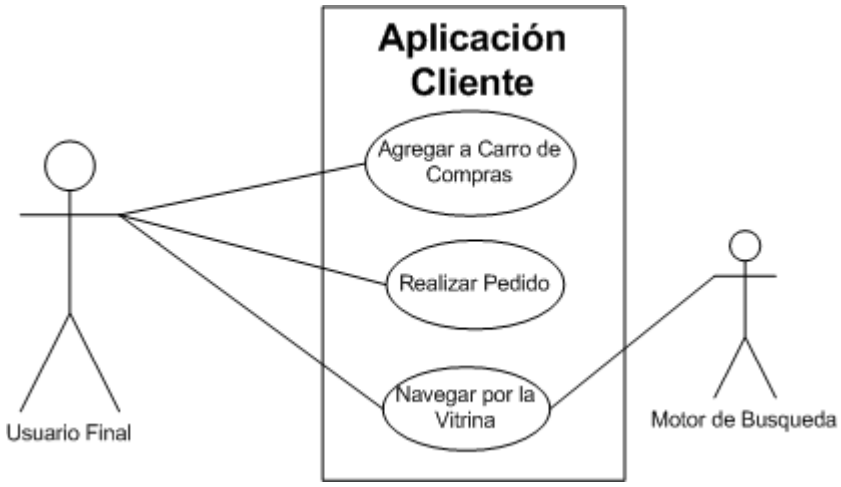

Figura 3-1: Casos de Uso en la Aplicación Cliente

Se realiza a continuación, una descripción detallada de los casos de uso de la aplicación cliente.

3.2.1 Caso de uso: Agregar a Carro de Compras

Actores: Usuario Final

Propósito: Agregar un ítem al carro de compras.

Precondiciones: Ninguna

Postcondiciones: El carro de compras aumentará en uno, el total de ítems del tipo que se adicionó.

Curso Básico de Acción:

1. El usuario final presiona el botón **agregar a carro de compras** de un plato, arrastra el plato al carro de compras o presiona el botón **aumentar cantidad del plato** en el carro de compras.
2. Si el ítem no se encuentra en la lista, se agrega al carro de compras con cantidad 1 y el caso de uso finaliza.
3. Si el ítem existe en la lista del carro de compras, la cantidad aumenta en 1.

Tipo: Secundario

3.2.2 Caso de uso: Realizar Pedido

Actores: Usuario Final

Propósito: Notificar al restaurante de la solicitud del pedido y guardar un registro de éste.

Precondiciones: El pedido debe tener un mínimo de dinero asociado, designado por el restaurante.

Postcondiciones: Se agregará una entrada a la base de datos, con el pedido y sus respectivos datos. La realización del pedido se confirmará vía mail, al cliente y al restaurante. El pedido estará disponible en la aplicación administrador, junto con la información asociada al pedido.

Curso Básico de Acción:

1. El cliente presiona el botón **realizar pedido**.
2. Se solicitará la fecha en la que requiere el envío.
3. Se le solicitará información adicional para poder concretar la entrega (nombre, teléfono, etc…).
4. Con toda la información validada, ésta será remitida al servidor.
5. En el servidor se agregará la información a la base de datos y se enviarán mails al usuario final y al administrador del restaurante.
6. Si hay éxito, se le informará al usuario que el pedido fue realizado.
7. Si no hay éxito, se entenderá que hubo un problema de comunicación y se le solicitará al usuario efectuar la operación nuevamente, en unos minutos.

Tipo: Primario

3.2.3 Caso de uso: Navegar por la Aplicación Cliente

Actores: Usuario Final, Motor de Búsqueda

Propósito: Exponer en el navegador la información del restaurante, categoría y plato.

Precondiciones: Ninguna

Postcondiciones: El actor obtiene la información, adaptada según sus requerimientos.

Curso Básico de Acción:

1. El cliente o el motor de búsqueda, solicitan una página al servidor.
2. Dependiendo del tipo de página solicitada, se devuelve información respecto del restaurante, categoría o plato en formato HTML.
3. En la situación en que el navegador no disponga del “plugin” de “Flash” o no ejecute código “javascript”, como es el caso de los motores de búsqueda y de algunos clientes, el caso de uso concluye aquí.
4. En caso de no contar con “Flash”, el contenido se mostrará en formato HTML y podrá ser indexado en los motores de búsqueda o ser visto en un “browser”. En caso de disponer del “Flash Player”, el contenido es remplazado por la película SWF.
5. La película SWF navega hasta la información solicitada.

Tipo: Primario

3.3 Casos de uso Aplicación Administrador

Dada la gran cantidad de casos de uso para esta aplicación, se desarrolló sólo los más complejos, dejando los más comunes y repetitivos para el Anexo B: Casos de Uso.

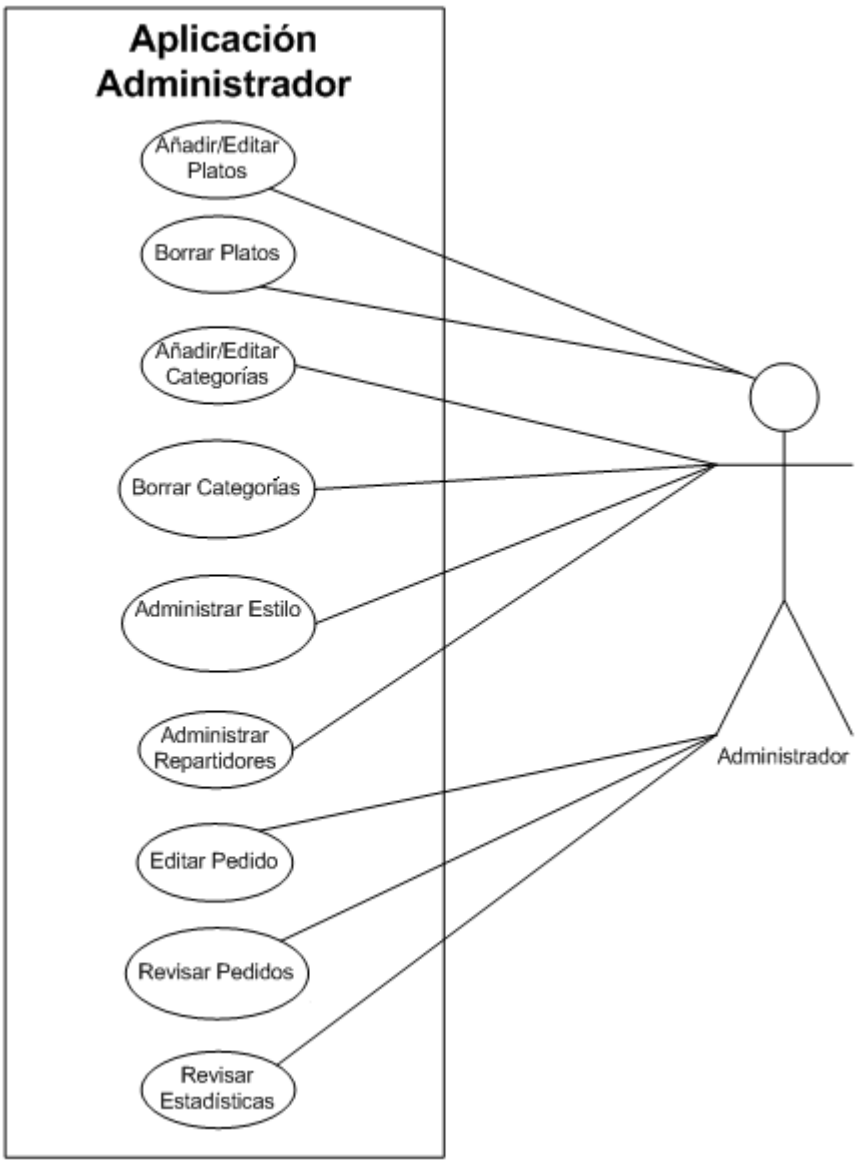

Figura 3-2: Casos de Uso de la Aplicación Administrador

3.3.1 Caso de uso: Administrar Estilo

Actores: Administrador

Propósito: Modificar el set de colores y distintos estilos en la aplicación cliente.

Precondiciones: Debe estar “logeado” y estar dentro de la sección de edición.

Postcondiciones: Las modificaciones se podrán ver en la aplicación cliente.

Curso Básico de Acción:

1. Al administrador se le presentarán los distintos campos, que podrá modificar.
2. Se podrán modificar los campos a libre elección.
3. Al presionar **aceptar**, se enviarán los cambios al servidor y se guardarán en la base de datos.

Tipo: Secundario

3.3.2 Caso de uso: Revisar Pedidos

Actores: Administrador

Propósito: Obtener una lista de los pedidos pendientes.

Precondiciones: Debe estar “logeado” y estar dentro de la sección de pedidos.

Postcondiciones: El restaurante tendrá una lista con todos los pedidos pendientes.

Curso Básico de Acción:

1. Se envía una solicitud al servidor de todos los pedidos con estado pendiente.
2. El servidor busca en la base de datos y retorna los registros asociadas.
3. Con la lista desplegada, se podrá dar paso al caso de uso “Editar Pedido”.

Tipo: Primario

3.3.3 Caso de uso: Editar Pedido

Actores: Administrador

Propósito: Modificar un pedido hecho por un cliente.

Precondiciones: Debe haberse llevado a cabo el caso de uso “Revisar Pedidos”.

Postcondiciones: Se modificará el pedido y se enviará un e-mail al cliente.

Curso Básico de Acción:

1. Se selecciona editar pedido, dentro de la sección de pedidos.
2. Se abrirá una vitrina dentro del módulo administración donde el administrador podrá navegar y añadir o quitar platos del carro de compras del cliente, de un modo similar a como lo haría desde la aplicación cliente.
3. Una vez efectuados los cambios, se enviará la nueva información al servidor y éste borrará la antigua entrada.

Tipo: Primario

3.3.4 Caso de uso: Revisar Estadísticas

Actores: Administrador

Propósito: Mostrar al restaurante las estadísticas esenciales de su negocio.

Precondiciones: Debe estar "logeado".

Postcondiciones: El restaurante verá las estadísticas representadas en un gráfico y podrá modificar ciertos parámetros, para ver como este fluctúa.

Curso Básico de Acción:

1. El administrador solicita información, asociada a un recurso (ventas, platos más solicitados, repartidores, etc.).
2. El servidor proveerá la información procesada.
3. La aplicación administrador, desplegará la información en gráficos dinámicos.

Tipo: Secundario

3.4 Diagramas de secuencia

Para entender de mejor manera los casos de uso comentados en la sección anterior, es necesario definir cuáles son las tareas de cada actor y cuándo se ejecuta cada una de ellas. Como la mayoría de los casos de uso se traducen en modificar información y realizar los cambios en la base de datos, se enfocará el resto del análisis en los casos de uso más complejos.

Para el curso básico de acción del caso de uso "Navegar por la Aplicación Cliente" se diseñaron dos diagramas. El primero, en la figura 3-3, muestra el caso de uso con un

usuario final como actor. El segundo muestra cómo se comportaría el sistema si el actor es un motor de búsqueda.

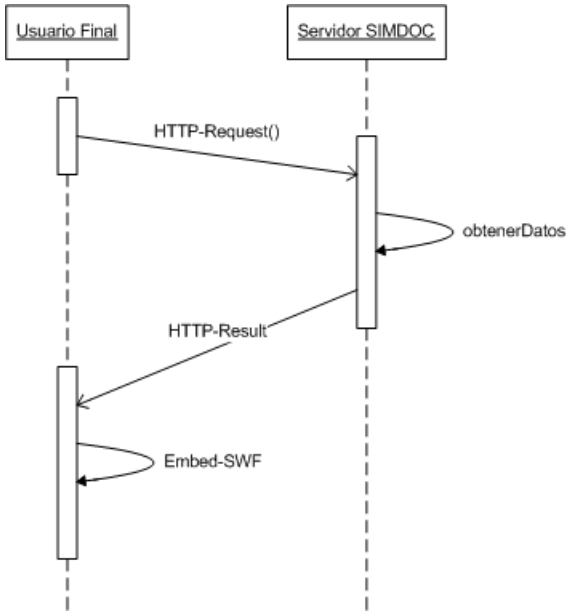

Figura 3-3: Diagrama de Secuencia “Navegar por la Aplicación Cliente (Usuario Final)”

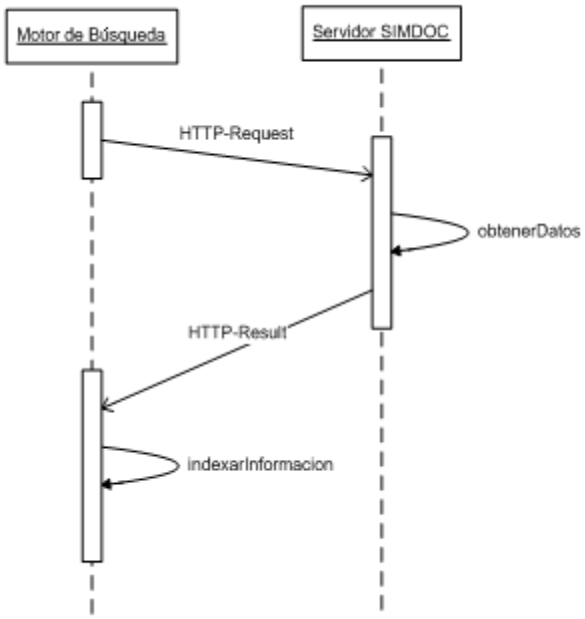

Figura 3-4: Diagrama de Secuencia “Navegar por la Aplicación Cliente (Motor de Búsqueda)”

En la figura 3-5 se combinaron los casos de uso “Revisar Pedidos” y “Modificar Pedidos” ya que este último tiene como prer-requisito la ejecución del primero.

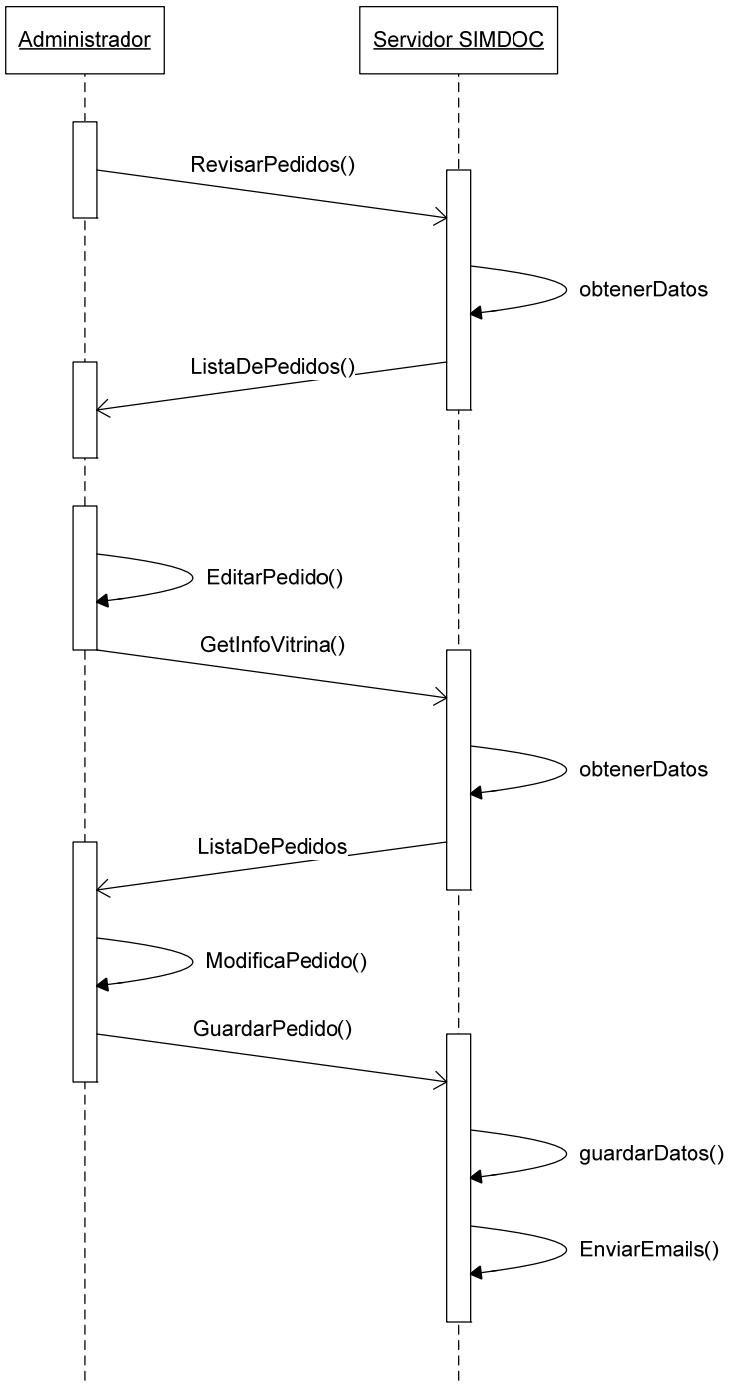

Figura 3-5: Diagrama de Secuencia para Revisar Pedidos y Modificar Pedidos

3.5 Diagramas de estado

Usando los casos de uso vistos en las secciones anteriores, se diseñaron las siguientes vistas para la aplicación cliente.

- **Vista Restaurante:** Suerte de portada, cuya función es mostrar slogan e imágenes del restaurante.
- **Vista Categoría:** Muestra todos los platos pertenecientes a una categoría.
- **Vista Plato:** Contiene un detalle del plato y sus posibles opciones.
- **Vista Pedido:** Expone un formulario para rellenar con datos personales y completar el pedido.

En la figura 3-6 se aprecia cada una de las vistas como un estado inactivo de la aplicación. Además, cabe destacar, que para limpiar el diagrama se omitieron las “escotillas de escape” cuyo rol es unir cada uno de los estados con el estado inicial, estas escotillas deben estar en la solución final ya que le dan al usuario, la seguridad que cada uno de sus acciones pueden ser canceladas sin afectar la integridad del sistema.

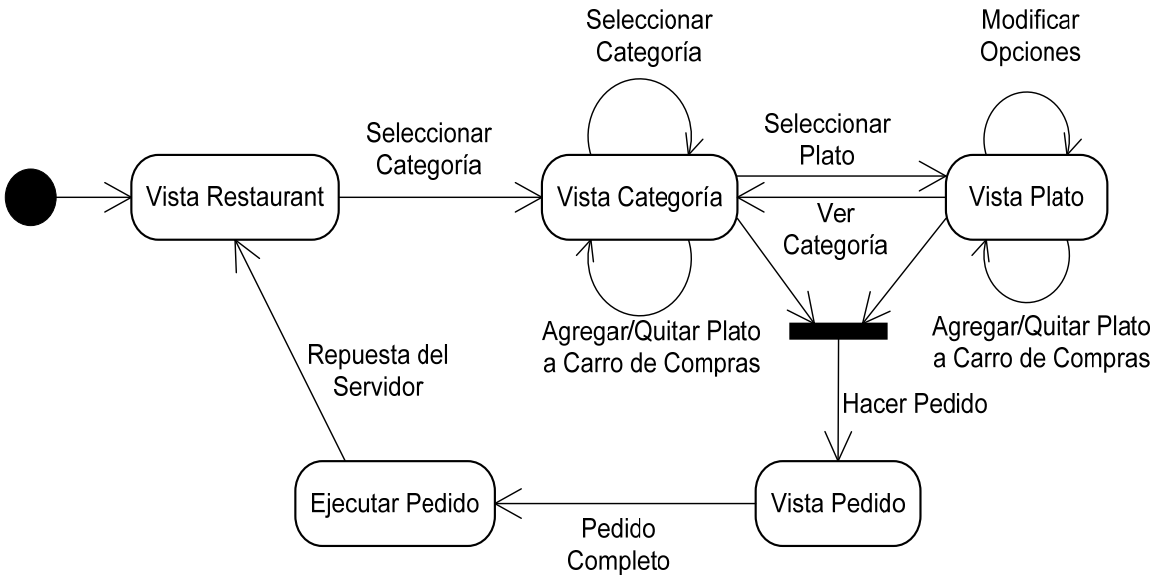

Figura 3-6: Diagrama de Estados para la Aplicación Cliente

Del mismo modo que la aplicación cliente, se generaron vistas para la aplicación administrador.

- **Vista Administrador Categorías:** Contiene una lista de las categorías existentes, aquí se pueden añadir y quitar categorías.
- **Vista Administrador Platos:** Muestra todos los platos dentro de una categoría seleccionada y es posible añadir y borrar estos.
- **Vista Administrador Opciones:** Aquí se observan todos los detalles de un plato.
- **Vista Administrador Pedidos:** Esta vista mostrará una lista de los pedidos solicitados.
- **Vista Administrador Repartidores:** Lista con la totalidad de los repartidores del restaurante.
- **Vista Administrador Repartidor:** Detalle de un repartidor en particular.

La aplicación administrador fue separada en 3 módulos, expuestos en figura 3-7, figura 3-8 y figura 3-9, para tener modelos más simples, cada uno de ellos deberá implementar el patrón de escotilla de escape.

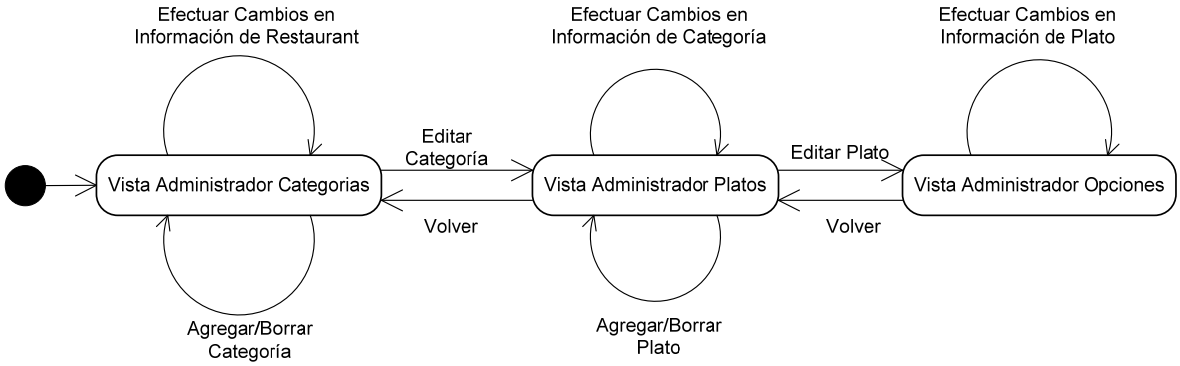

Figura 3-7: Diagrama de Estados para la Administración de Categorías y Platos

La administración de categorías y platos tiene sólo tres estados, correspondientes a cada uno de los niveles de la aplicación cliente. Este tipo de diseño podría llevar, tanto a un tipo de edición por formas y campos, o a un formato de edición cercano al paradigma

WYSIWYG, donde el usuario se enfrenta a la vista final del ítem, en vez de campos de texto u otro tipo de controles de entrada.

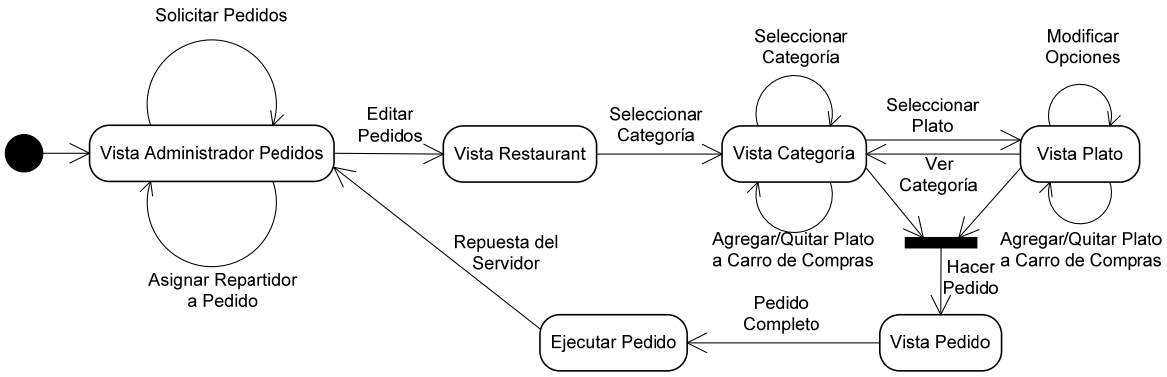

Figura 3-8: Diagrama de Estados para la Administración de Pedidos

Cabe distinguir la similitud entre el diagrama de administración de pedidos y la aplicación cliente, siendo diseñado de esta manera para mantener una concordancia entre ambos sistemas y la forma en que interactúan con el servidor, además de acortar el tiempo de familiarización del usuario con el sistema.

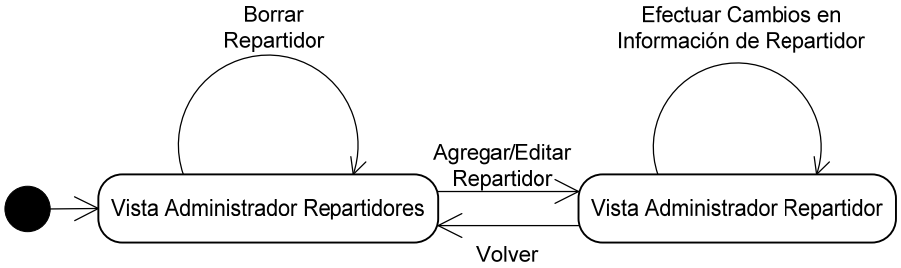

Figura 3-9: Diagrama de Estados para la Administración de Repartidores

El módulo de administración de repartidores es uno de los más pequeños; sin embargo, es el primer paso para enfocar el sistema completo hacia el área de “delivery” de comidas.

3.6 Arquitectura

La modularidad de una aplicación, es una de las características necesarias para que el proyecto permanezca y perdure en el tiempo. Para lograr esto, no sólo es necesario que el código sea simple y esté dividido en pequeñas clases, sino que es conveniente también, delimitar distintos niveles de abstracción donde cada capa sea responsable de una o dos tareas a lo sumo, de modo que las clases resultantes sean fáciles de leer y modificar. De esta manera, se pueden realizar optimizaciones en una capa sin alterar el funcionamiento de las demás.

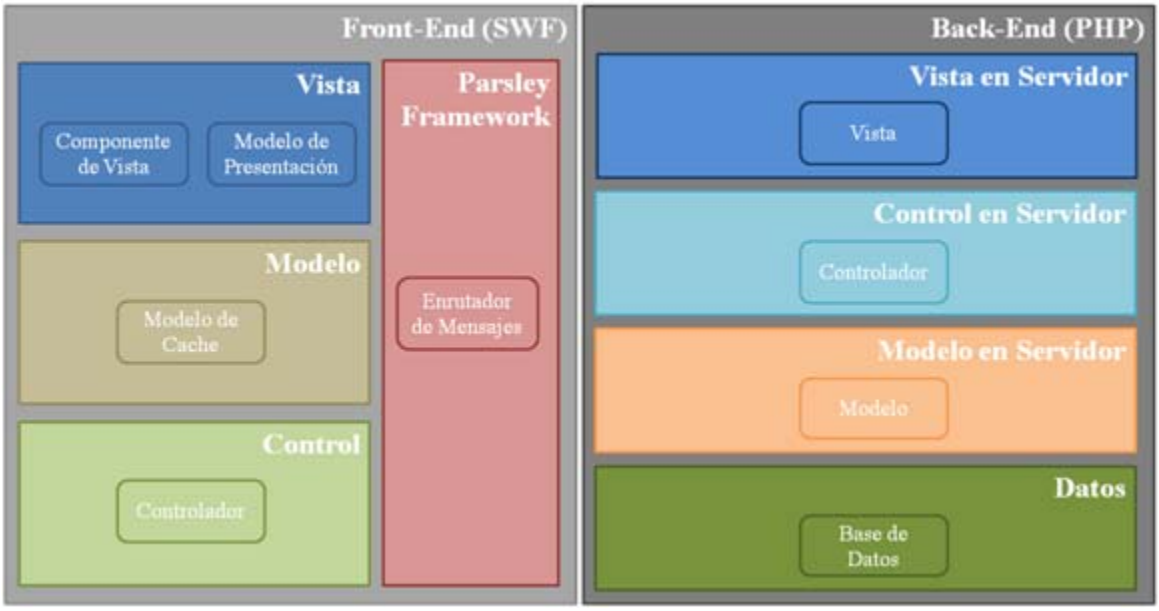

Figura 3-10: Arquitectura del Sistema

A continuación, se describirán las responsabilidades de cada componente de la arquitectura expuesta en la figura 3-10:

- **Vista:** Encargada de desplegar la información de la capa **Modelo** e informar a la capa **Control** sobre la interacción del usuario con la aplicación.
    - **Modelo de Presentación:** Éste subcomponente de la **Vista** contiene toda la lógica asociada a ésta, tiene acceso directo al **Modelo** y es capaz de enviar eventos al **Control**. Se programa típicamente en lenguaje Action Script.

    - **Componente de Vista:** Este subcomponente no contiene lógica y obtiene la información a ser mostrada del **Modelo de Presentación**. Este componente se programa en MXML y su desarrollo puede ser derivado a un diseñador, al carecer de lógica ni acceso a otras partes de la aplicación.
- **Modelo:** Mantiene en caché la información a desplegarse, además de informar a la capa **Vista** sobre eventuales cambios.
    - **Modelo de Caché:** Es el subcomponente básico del **Modelo**. Por lo general, tiene una contraparte en el **Modelo en Servidor** aunque no es indispensable.
- **Control:** Contiene la lógica de la aplicación, responde a eventos gatillados por el usuario y ejecuta cambios en el modelo y en la vista, de ser necesario solicita información extra al **"Back-End",** el **"Framework Parsley"** es el encargado de manejar los resultados.
    - **Controlador:** Es la unidad básica del **Control** y está programado en Action Script.
- **"Framework Parsley":** Se preocupa de la comunicación entre las distintas capas para aminorar las dependencias entre las clases, haciendo posible cambios en las clases de cada una de las capas, sin necesidad de alterar el código de clases relacionadas.
    - **Enrutador de Mensajes:** Es el componente del **"Framework Parsley"** encargado de administrar los eventos y asegurarse que estos lleguen a su destino.
- **Vista en Servidor:** Para el caso en que motores de búsqueda, usuarios sin Flash o usuarios en dispositivos móviles, naveguen por la página, esta capa mostrará la información que debería verse en la aplicación, sin involucrar nada de la lógica.
    - **Vista:** El componente básico de la **Vista en Servidor**, está escrito preferentemente en PHTML y no contiene mayor lógica que la inserción directa de las variables generadas por el **Control en Servidor**. Este archivo puede ser derivado a un diseñador sin que necesite alterar el código PHP.
- **Control en Servidor:** Recibe las solicitudes del "web browser" y del "Flash Player", es capaz de interpretarlas y generar una vista, en el caso de que sea una solicitud HTML, o devolver un objeto, en el caso que el mensaje sea AMF.

- **Controlador para AMF:** Encargado de recibir las consultas del Flash Player. Si bien el archivo es una clase PHP, la respuesta es convertida al formato AMF antes de ser remitida.
- **Controlador para Web:** Contiene toda la lógica necesaria para generar una página web usando los "scripts" de la **Vista en Servidor**.

- **Modelo en Servidor:** Esta capa es una representación en el servidor del modelo que se encuentra en el cliente, además tiene acceso a todas las operaciones CRUD de la base de datos.[11]
  - **Modelo:** Esta clase por lo general es la contraparte del **Modelo de Caché** en el "**Front-End**" y es preferible, además, que represente una fila de una tabla de la base de datos y no un objeto con mayor complejidad.
- **Datos:** Guarda la información completa del sistema y consta de un motor que cumple con las propiedades ACID.[12]
  - **Base de Datos:** En este caso se usará el motor MySQL.

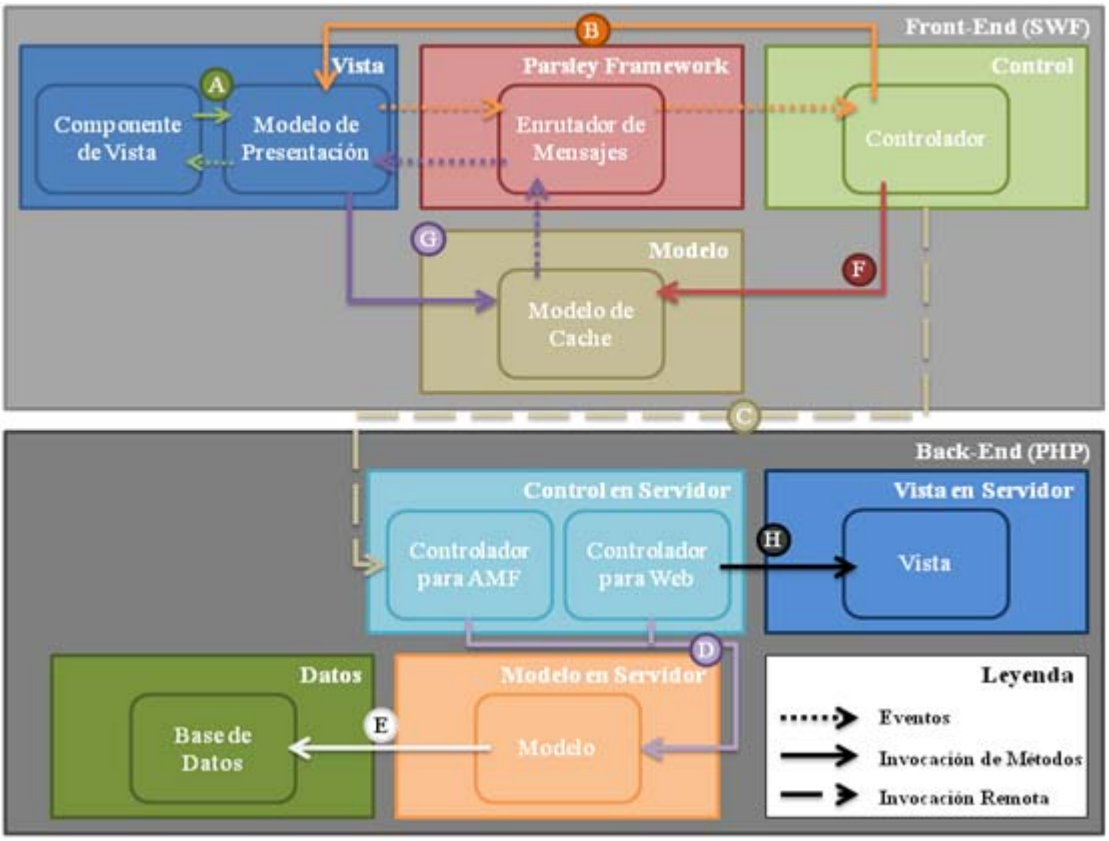

Figura 3-11: Comunicación entre Componentes

Las interacciones en el diagrama de la figura 3-11 son descritas a continuación.

A. El **Componente de Vista**, es capaz de invocar métodos del **Modelo de Presentación** para obtener información a desplegar o despachar eventos al resto de la aplicación. En el momento en que en el **Modelo de Presentación** se produzca un cambio, éste debe enviar un evento al **Componente de Vista** para que se expongan al usuario los cambios realizados.
B. El **Modelo de Presentación** despacha la interacción del usuario al **Controlador** en forma de evento, para que éste actúe de acuerdo al tipo de evento. El **Controlador** por su parte ejecuta métodos en el **Modelo de Presentación**.
C. Si se requiere lograr un cambio en la base de datos u obtener información adicional, el **Controlador** invocará a un **Controlador para AMF**.
D. Tanto el **Controlador para AMF** como el **Controlador para Web** generan distintos **Modelos en el Servidor**, para poder trabajar.
E. Los **Modelos en Servidor** se generan a partir de la información guardada en la **Base de Datos**.
F. El **Controlador**, una vez que tiene la respuesta del servidor, altera directamente la información del **Modelo de Caché**.
G. El **Modelo de Caché** envía un evento de actualización al **Modelo de Presentación**, y ésta, por su lado, obtiene la información necesaria del **Modelo de Caché**.
H. Al ser solicitada una página web, el **Controlador para Web** instancia una **Vista en servidor** y usando los **Modelos en Servidor** generados en el punto D asigna valores a las variables, que la **Vista** utilizará para crear el código HTML a entregar.

4 Implementación del sistema propuesto

Con el diseño y arquitectura de la solución ya descritos, se procede a la implementación del sistema completo. En este capítulo, se analizarán los componentes principales del proyecto y la forma en que se desarrollaron.

4.1 Implementación en el servidor

En el servidor se trabajan escenarios de capas más bajas, donde se crean soluciones que conforman una base estructural, para la aplicación cliente. Las funcionalidades más importantes en esta área, fueron la comunicación con el cliente y la indexación del contenido, en motores de búsqueda.

4.1.1 Comunicación con el cliente

En el mundo de las RIAs existen 3 lenguajes de transferencia de datos. El más antiguo y más conocido, es XML. Este lenguaje tiene la ventaja de haber sido desarrollado para organizar documentos, puede usar esquemas para validar el archivo y así evitar problemas en la comunicación entre el cliente y el servidor (en el caso que haber sido desarrollados por equipos distintos). Sin embargo, al trabajar con XML, los archivos resultantes pesan más de lo estrictamente necesario dado su constante uso de “tags” de abertura y cierre. El tamaño del paquete crece en una medida mucho mayor que el tamaño de información a ser enviada, debido a la redundancia innecesaria, lo cual constituye una desventaja importante en el momento de decidir por cual optar.

El siguiente lenguaje a conocer es JSON, desarrollado para “javascript”. Es bastante más ligero que XML ya que le quita toda su redundancia. Si la aplicación que se está desarrollando está basada en AJAX, lo más probable es que ésta sea la elección natural.

AMF por su lado, es un protocolo binario desarrollado por Adobe y tiene el mismo formato que usa el “Flash Player” para sus objetos en tiempo de ejecución, lo que reduce significativamente los tiempos de decodificación.

Evaluando los pros y contras de cada lenguaje se decidió trabajar con AMF ya que la comunicación binaria es la más ligera de las 3, además que Flex lo trabaja nativamente, por lo que se evita pasar por procesos de decodificación.[13]

Mayor información sobre la ejecución de servicios remotos usando AMF se puede encontrar en el Anexo C: Ciclo de Vida de una Llamada Remota

4.1.2 Indexación en motores de búsqueda

Si bien es sabido que la indexación de contenido dinámico de una película SWF no es posible, tampoco lo es la indexación del contenido de una imagen. Los motores de búsqueda, sin embargo, usan las palabras claves escritas en el campo ALT del tag <IMG> del lenguaje XHTML para asociarlas a la imagen.

De este modo, el servidor generará una vista completa en XHTML, con la misma información que el usuario vería en la película SWF; al momento de cargarse la página se ejecutará un script que reemplazará todo el contenido dentro del tag <DIV> por el objeto SWF. Quien vea el objeto SWF tendrá una experiencia más rica con el contenido la página; quien no cumpla las condiciones para ejecutar la película SWF verá el mismo contenido en formato HTML. De esta forma el robot descargará la página con todo el contenido y la indexará de acuerdo a las palabras clave que encuentre en ésta.

Luego de asegurar que el contenido es expuesto a los robots de los buscadores, se procede a diseñar el sitio, teniendo en cuenta las recomendaciones “White Hat SEO”, que es un conjunto de buenas prácticas para que la navegación por el sitio y la indexación de palabras claves, sea de manera sencilla y natural. Además se deben evitar las practicas “Black Hat SEO”, que tienden a forzar las palabras claves a la base de datos de los buscadores. Esta últimas son gravemente penalizadas por los motores de búsqueda. Más detalles sobre “White Hat SEO” y “Black Hat SEO” se encuentran en el Anexo D: Sombrero Blanco contra Sombrero Negro.

Las prácticas seleccionadas y usadas en el diseño del sitio son: [14][15][16][17][18]

- Links internos: Cada página del sitio tiene links de navegación local, dentro de cada restaurant y un link a la página principal.
- Palabras clave como “links”: Cada enlace, debe contener un texto descriptivo sobre la página a la que el “link” apunta, de esta forma los buscadores tendrán una idea no sólo del contenido de una página, sino de qué formas, otras páginas hacen link a ésta.
- Título únicos y descriptivos: Cada página del sitio, consta de un título único generado de la siguiente manera:

  Nombre del Restaurant – Nombre de la Categoría – Nombre del Plato

- Uso del metadato **“description”**: Cada página del sitio tiene su propia descripción, que no es vista por el usuario, sólo le sirve al buscador para generar resultados de búsqueda.
- Nombres descriptivos de las imágenes: Los nombres de las imágenes del sitio tienen directa relación con lo que representan, de este modo no sólo podrán ser bien indexadas sino que además al ser mostradas como resultado de búsqueda, serán entendibles por un usuario cualquiera.
- Uso del parámetro ALT en las imágenes: Cada imagen tiene su propio texto alternativo, relacionado con lo que se muestra.
- URLs descriptivas: Al usar páginas dinámicas, un desarrollador puede vers inclinado a usar los parámetros GET del protocolo HTML del siguiente modo:

  http://simdoc.cl/index.php**?idRestaurant=25&idCategoria=30&idPlato=15**

  Sin embargo, esto es considerado mala práctica por los buscadores (a pesar de que con un buen mapa del sitio se pueden indexar todas las páginas dinámicas).

  Un mejor acercamiento es el uso de URLs más organizado y cercano al lenguaje humano de esta forma:

http://simdoc.cl/**nombre-del-restaurant/nombre-de-la-categoria/nombre-del-plato**

Esta forma de diseñar una URL resulta más descriptiva en cuanto a qué es lo que se está observando, además de ser más fácil de recordar por usuarios del sistema. Este mismo acercamiento está diseñado, para las direcciones de las imágenes del sitio.

- Uso de mapas del sitio: En el sitio http://www.sitemaps.org/ existe una documentación completa sobre cómo usar el protocolo “Sitemap”, para dar información adicional sobre cada página del sitio.

4.1.3 Base de Datos

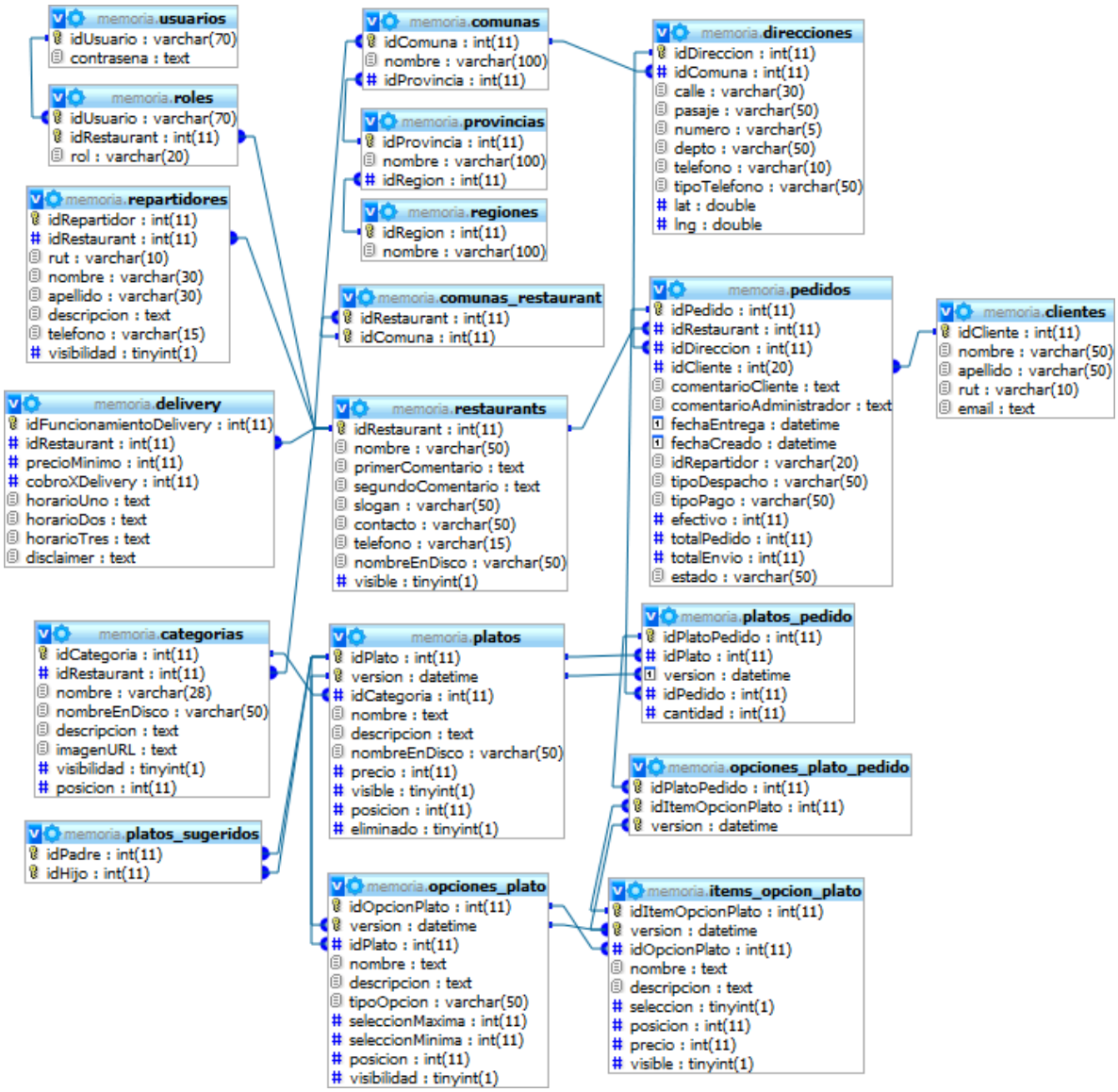

Figura 4-1: Diagrama de la Base de Datos

4.2 Implementación en el cliente

Las implementaciones de las aplicaciones tanto para el administrador como para el cliente, están enfocadas a capas más altas. Desde el punto de vista de la aplicación cliente, es necesario dividir las opciones que tenga el usuario para no presentarle mucha información al mismo tiempo. En los siguientes puntos se verán cómo se dividieron las aplicaciones cliente y administrador.

4.2.1 Aplicación Cliente

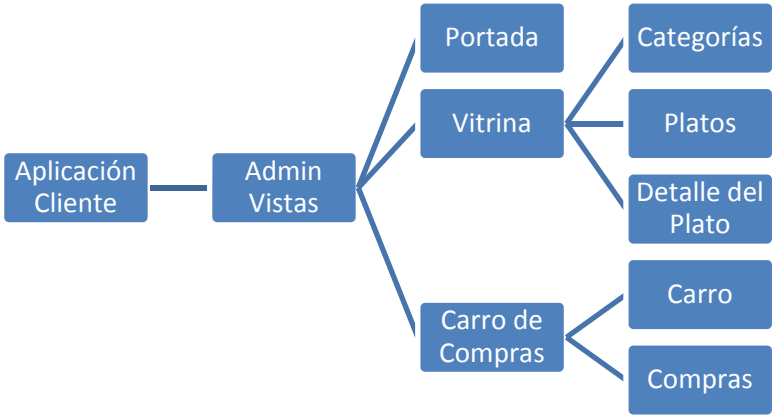

Figura 4-2: Jerarquía de Módulos de la Aplicación Cliente

La figura 4-2 expone cada una de las vistas a las que podrá acceder un visitante en la página del restaurant. En primer lugar, se tiene el módulo de aplicación cuyo único rol es inicializar los modelos, vistas y controladores. Luego en la jerarquía se visualiza el módulo de administración de vistas, cuya función es decidir cuál de todas las vistas debe mostrar al usuario.

La portada es un pequeño módulo, cuya única obligación es presentar comentarios de bienvenida a la página, adaptables según la necesidad del administrador.

La vitrina consta de dos secciones. La primera, siempre visible, es la vista de categorías, que sirve para navegar a través de las distintas categorías que ofrezca el

restaurant. La segunda sección de la vitrina, puede ser ocupada por: la vista de platos, cuya misión es mostrar el catalogo de los platos de una categoría; o la vista de detalle del plato, que ofrece información adicional del plato y posibles opciones modificables del mismo.

Finalmente, se tiene el módulo del carro de compras que consta de 2 fases. La primera es la del carro en sí, donde se puede agregar o quitar platos. La segunda fase, llamada compras, comprende la vista donde se procede a concretar el pedido y rellenar los datos personales requeridos.

4.2.2 Aplicación Administrador

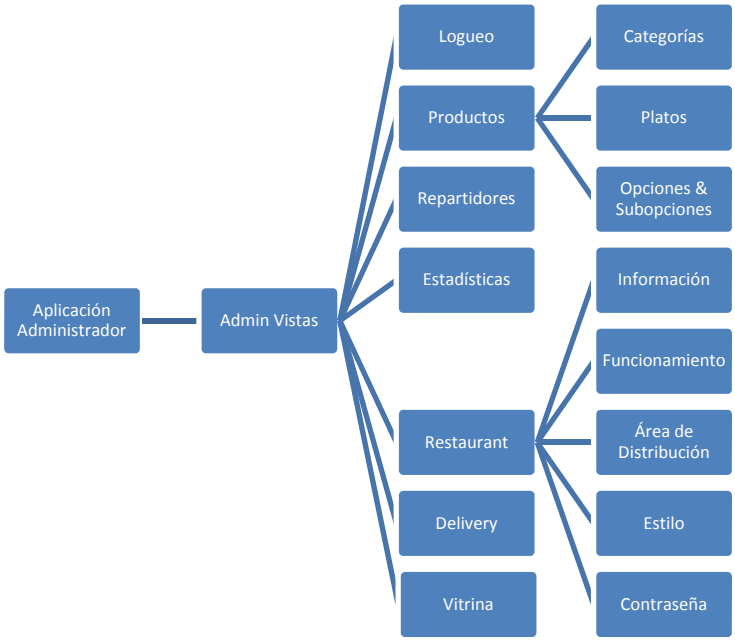

Figura 4-3: Jerarquía de Módulos de la Aplicación Administrador

Del mismo modo que en el punto 4.2.1, en la figura 4-3 se puede apreciar que la aplicación administrador consta de los módulos de aplicación y administrador de vistas. Estos módulos tienen las mismas tareas, que en la aplicación cliente.

El módulo de logueo es el primero en desplegarse, ya que la aplicación desconoce qué restaurante deberá administrar. Esta información se deducirá del par usuario contraseña ingresado por el usuario.

El módulo de productos, donde el administrador ingresará cada uno de los productos que desea proveer, consta de 3 vistas. La primera es la de las categorías, donde se podrán modificar los nombres y descripciones de éstas, además de la posición en que se mostrarán. Junto a las categorías, se encuentra la vista de los platos que pertenecen a cada categoría, aquí es donde se editan los detalles de cada plato. Si un plato es más complicado y ofrece distintas opciones para ser pedido, se puede acceder a la vista de opciones. Dentro de esta vista, se muestran las opciones del plato junto con sus subopciones, que son los posibles valores que podrá adquirir cada opción. Por ejemplo el plato “Pollo con agregado” presenta una opción llamada “Agregado” que a su vez tiene dos subopciones “Papas Fritas” y “Puré”, el comprador tendrá que escoger una de estas subopciones que podrán eventualmente, alterar el valor total del plato.

El siguiente módulo es el de repartidores, lugar en el cual el administrador podrá llevar un registro de los empleados, encargados de entregar los pedidos a los consumidores.

En el módulo de estadísticas, se podrá acceder a los distintos registros relacionados con los pedidos efectuados al restaurant; pudiendo revisar las ventas totales, la demanda de los platos y el desempeño de los repartidores.

El módulo de restaurant, es el que más vistas contiene, ya que acá se encuentra la funcionalidad básica del restaurant. La primera vista es la de información del restaurant, lugar que puede modificar variables propias de la identidad del restaurant: nombre, comentarios, email y teléfono de contacto. En la vista de funcionamiento se trabajan variables del negocio de “delivery” que llevará el restaurant, tales como: horarios de atención, cobro por reparto, etc. En la vista de área de distribución, cada restaurant podrá inscribir las comunas a las cuales repartirá. En la base de datos se consideran todas las comunas de Chile. El cambio de logo, imagen de portada y fondo, podrán realizarse en la vista de estilo. La contraseña del administrador, puede ser cambiada en la vista de contraseña.

La vista de “delivery”, tiene acceso a una lista con todos los pedidos realizados por los consumidores de ese restaurant, donde el administrador podrá editar estos pedidos, asignarles un estado de entrega y elegir qué repartidores se harán cargo de ellos.

Por último, el módulo vitrina muestra al administrador una vista igual a la aplicación cliente. Dentro de ésta, el administrador podrá modificar el contenido de los pedidos en caso de ser necesario. Este módulo funciona del mismo modo que la aplicación cliente, por lo que no debería ofrecer mayor dificultad al usuario.

5 Pruebas de funcionamiento y rendimiento del sistema

Es de absoluta importancia el correcto funcionamiento e implementación del sistema propuesto. Para esto, se realizó una serie de pruebas que analizaron diferentes aspectos del comportamiento y del desempeño del sistema. Estos procedimientos incluyen tanto pruebas de funcionamiento como pruebas de rendimiento.

5.1 Pruebas de uso

En la primera fase se tienen como objetivos, por un lado, comprobar que se realicen las tareas para las cuales fue diseñado el sistema, y por otro, confirmar que efectivamente cumple con los requisitos enumerados en el capítulo 2, sección 2.2. A continuación se detallan las metodologías utilizadas y los resultados obtenidos tras validar y verificar el sistema desarrollado.

5.1.1 Metodología

Mientras se fue desarrollando el proyecto, se realizaron pruebas del tipo “unit testing”, estas son pruebas que se efectúan a cada módulo una vez concluida su elaboración. Estas tienen como objetivo el detectar posibles errores, antes de que estas piezas estructurales sean incorporadas a la aplicación final. Los módulos pueden dividirse en dos tipos: los pertenecientes al servidor y los pertenecientes a las aplicaciones.

En el caso de las aplicaciones se utilizó el modo “debug” del IDE “Adobe Flash Builder 4” (herramienta con la cual fueron desarrolladas), para verificar que efectivamente la información entregada y manejada por un módulo, era la deseada. Por ejemplo: en la aplicación cliente, específicamente en el módulo del carro de compras, se verificó que la información que se entrega al servidor al realizar un pedido, esté conforme a lo que se espera. Es decir, que los platos y las opciones que fueron seleccionadas por el cliente y que

la información personal y de despacho que fueron introducidas por el mismo, efectivamente lleguen correctamente al servidor.

Se adquirió una aplicación llamada “Service Browser” que permite probar el correcto funcionamiento de servicios montados en un servidor que trabaja a base del framework AMFPHP. Como el proyecto se realizó utilizando el “Zend Framework”, hubo que realizar diversas modificaciones a “Service Browser” para que pudiese funcionar con el “framework” empleado. Cabe mencionar que también se tuvo que realizar algunos cambios en el “Zend Framework” para que éste detectase cuándo era “Service Browser” el que efectuaba una llamada a los servicios y no una de las aplicaciones directamente relacionadas con el proyecto. Esto con el fin de que le entregase información adicional para su posterior análisis. Cada vez que concluyó el desarrollo de un servicio, se utilizó esta aplicación para probar que efectivamente el servicio entregara la información que era necesaria. Se debe resaltar también, que no todos los servicios pudieron ser probados de esta manera, algunos debieron ser testeados utilizando el modo “debug” del IDE mencionado anteriormente, ya que los argumentos de entrada de algunos servicios impedían el uso de “Service Browser”.

Una vez concluido el desarrollo y el “unit testing” de cada módulo, se procedió a integrarlos en una sola aplicación final, tanto la de la aplicación cliente, como la de la aplicación administrador. Ya ensamblado, se procedió a realizar pruebas en base a los diversos casos de uso existentes. Se probó absolutamente cada una de las funcionalidades que se entregaba al usuario, y se verificó que se realizaran correctamente.

Luego de haber concluido estas pruebas, se dispuso a entregar la aplicación cliente a dos usuarios con conocimientos computacionales básicos y la aplicación administrador a dos usuarios de nivel medio. Esto se hizo con el propósito de analizar su respuesta frente a las aplicaciones y la usabilidad y comprensión de la solución propuesta. Se consideraron como usuarios básicos aquellos que hacen uso del computador sólo para acceder a las redes sociales, hacer uso de sus correos, realizar compras por Internet, etc. En cambio, se tuvieron como usuarios medios aquellos que utilizan computadores para trabajar y hacen uso de diversos programas en el mismo, por ejemplo: Microsoft Outlook, Microsoft Excell, programas para áreas específicas, etc. Primero se les entregará el funcionamiento de un

sistema de delivery y luego se les hablará acerca de las funcionalidades con las que cuenta este sistema. La idea es verificar si ambos pueden hacer uso de la mayoría de las funciones sin problema alguno. Debe tomarse en cuenta que ninguna de estas personas se familiariza con el proceso de delivery, por lo tanto necesitarán una breve introducción al tema. En caso de ser necesario se les entregará información adicional. Se les hizo participar de los siguientes casos de uso como: gestionar un pedido, ver las estadísticas de los pedidos, crear/modificar/eliminar un plato.

5.1.2 Resultados obtenidos

Los resultados obtenidos fueron exitosos. Se comprobó el correcto funcionamiento de cada uno de los módulos y servicios por separado, y de todo el sistema, una vez ensamblado.

El test de usabilidad, comprensión y respuesta realizado con los cuatro usuarios a los que se les entregó el software fue favorable. En el caso de la aplicación cliente los dos usuarios no necesitaron de ayuda extra para realizar sus pedidos y para seleccionar los platos con las opciones que desearon, lo cual demuestra que la aplicación es intuitiva. En cuanto al uso de la aplicación administrador el primer usuario no tuvo necesidad de ninguna instrucción adicional para completar exitosamente los casos de uso, sin embargo, no ocurrió lo mismo con el segundo usuario. Este último completó exitosamente los casos de ver las estadísticas de los pedidos y crear/modificar/eliminar un plato, pero no pudo gestionar un pedido de forma independiente ya que requirió instrucciones más detalladas para poder lograrlo.

5.2 Pruebas de desempeño

En la segunda fase de pruebas, se analizó el comportamiento del servidor en diversos escenarios. Se buscó obtener el rendimiento del equipo en situaciones normales y de alta demanda. Como este es un sistema que anhela abarcar un número importante de empresas del área gastronómica, se debió demostrar que era capaz de responder frente a situaciones de alta carga, ya que a ciertas horas del día (ej: la hora de almuerzo) sufrirá de constantes episodios en los que se demandarán una alta cantidad de recursos de su parte.

Cabe destacar, que las pruebas para medir el rendimiento de aplicaciones web de alta demanda que cumplen con los criterios utilizados hoy en día, no son posibles de realizar en un proyecto de examen de pregrado. Esto se fundamenta en que se requiere de un software especializado y tener capacitación en el mismo, lo cual se traduce en dos cosas: gastos importantes de dinero y de tiempo, con los que no se cuenta para realizar un proyecto de esta envergadura. Por lo anterior, se decidió desarrollar una metodología apropiada para un proyecto de este tipo. En el Anexo F: Estrategia de Prueba de Rendimiento para SIMDOC se detalla una segunda metodología desarrollada que intenta satisfacer dichos criterios.

5.2.1 Metodología

La medida más importante al momento de analizar el rendimiento del sistema, es el tiempo de respuesta de los servicios. Esto se refiere al tiempo que transcurre entre que el cliente efectúa una llamada a un servicio y lo que le toma a la respuesta llegar hasta el cliente. Otras medidas también importantes, son el uso de CPU y de memoria RAM por parte del servidor, al atender un cierto número de clientes actuando simultáneamente. La última medida que se tomó en cuenta, fue la influencia que tuvo el ancho de banda del servidor en cuanto al rendimiento del mismo frente a situaciones de stress.

Se estudiaron dos escenarios, que difirieron sólo en cuanto al ancho de banda del servidor. En el primer escenario, el servidor contó con un ancho de banda de 100 Mb/s tanto para el “downlink” como para el “uplink”. En el segundo, se le entregó al servidor un ancho de banda de 10 Mb/s para el “downlink”, y 10 Mb/s para el “uplink”. Para cada escenario se varió el número de clientes simultáneos que se conectaba al servicio, desde ahora sub-escenarios. Se realizaron 10 mediciones por cada escenario, iniciando con 1 cliente hasta llegar a 10 clientes simultáneos.

En base a estos escenarios se distinguió la importancia que tiene el ancho de banda del servidor frente al rendimiento del mismo. Los sub-escenarios, ayudaron a obtener el tiempo de respuesta de los servicios, el uso de CPU y de memoria RAM del servidor mientras un cierto número de clientes, interactuaba con él.

Es importante mencionar que si bien el servidor cuenta con alrededor de 100 servicios, sólo se utilizó el servicio llamado “getItems” (perteneciente al módulo de la aplicación cliente en el servidor) para realizar las mediciones en ambos escenarios. Lo anterior se debe a que éste, es uno de los servicios que más trabajo realiza en el servidor, ya que se revisan las tablas más importantes de la base de datos y se obtienen los datos necesarios para el completo funcionamiento de la aplicación cliente y administrador. Sólo algunos servicios, son comparables al recién mencionado en términos de trabajo en el servidor y es por lo mismo que se ha elegido realizar estas pruebas con getItems.

En la figura 5-1 se detalla el diagrama de red usada para realizar el análisis.

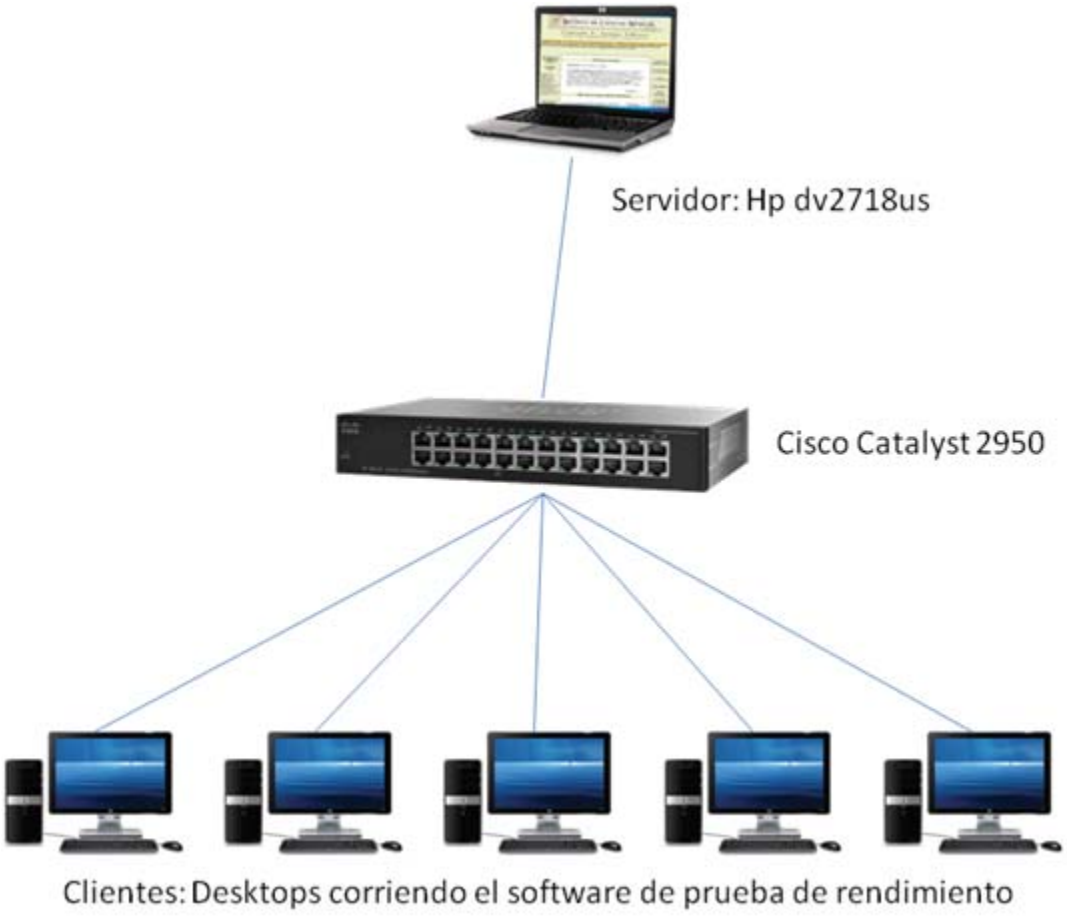

Figura 5-1: Diagrama de red utilizado para ambos escenarios

Todos los enlaces que se encuentran en la figura anterior son Full-Duplex, y por defecto todos trabajan a 100 [Mbps]. El número de Desktops conectados al switch varía en cada sub escenario.

El equipo que hizo el papel de servidor es un notebook modelo Hp dv2718us, que cuenta con un procesador AMD Turion 64 X2 de 2.1 [GHz] y con 3 [GB] de memoria de RAM. Funciona con Windows 7 y posee una tarjeta de video integrada. El ancho de banda del enlace entre el switch y el servidor, se controlará desde la tarjeta de red de este último. Cabe resaltar, que esta línea de notebooks desarrollada por Hp sufre de un problema de diseño, los equipos no ventilan como corresponde y por lo tanto se sobrecalientan, disminuyendo así su eficiencia en cuanto a procesamiento.

Los Desktop son equipos que funcionan trabajan en Windows XP Profesional, Service Pack 2. Además cuentan con un procesador Intel Core Duo 1.6 [GHz] y con 504 [MB] de memoria RAM.

Se desarrolló un software especialmente para realizar las pruebas, el cual se encarga de realizar 100 llamadas a un servicio llamado getItems en el servidor. Por cada llamada se mide diferentes tiempos relacionados con el servicio, como por ejemplo: tiempo que pasó en la red, tiempo de codificación y decodificación, tiempo que toma la realización del servicio en el servidor, etc. Una vez que se han completado las 100 llamadas, la aplicación calcula el promedio de todas las llamadas, por cada campo de tiempo medido. Al comenzar cada experiencia se corrió esta aplicación y se iniciaron simultáneamente las mediciones de cada una de ellas.

En cuanto a las mediciones del uso de CPU y de memoria RAM, estas se midieron usando un gadget de Windows 7 llamado “Core Temp”, el cual indica las cargas y las temperaturas de los núcleos, la cantidad de la memoria utilizada en porcentaje y en [MB].

5.2.2 Resultados obtenidos

A continuación se detallan los gráficos de los resultados obtenidos en la prueba de rendimiento, los que luego serán analizados:

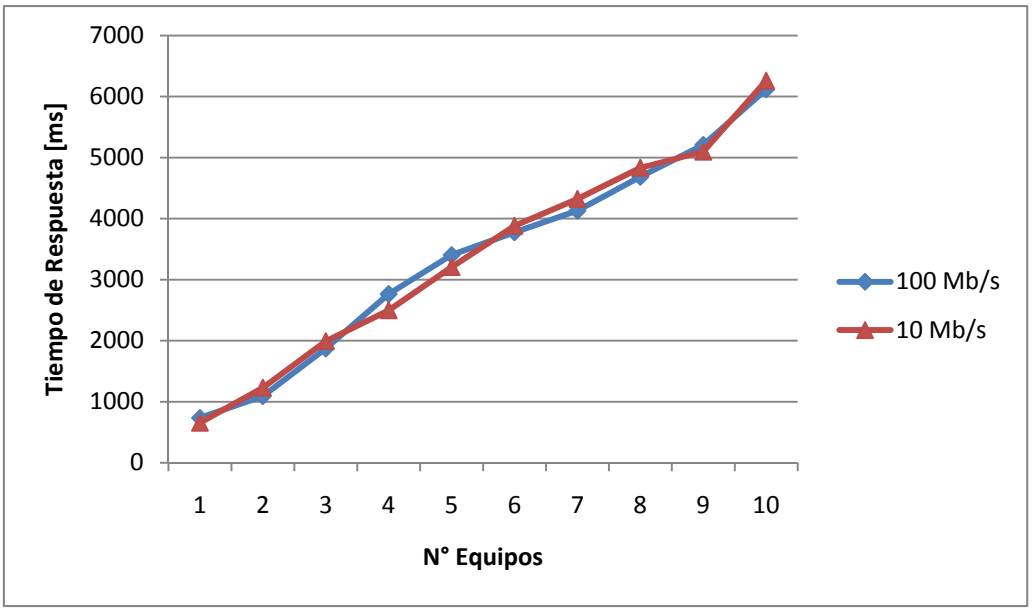

Figura 5-2: Tiempo de Respuesta del Servicio “getItems” v/s el Número de Equipos efectuando Peticiones al Servidor

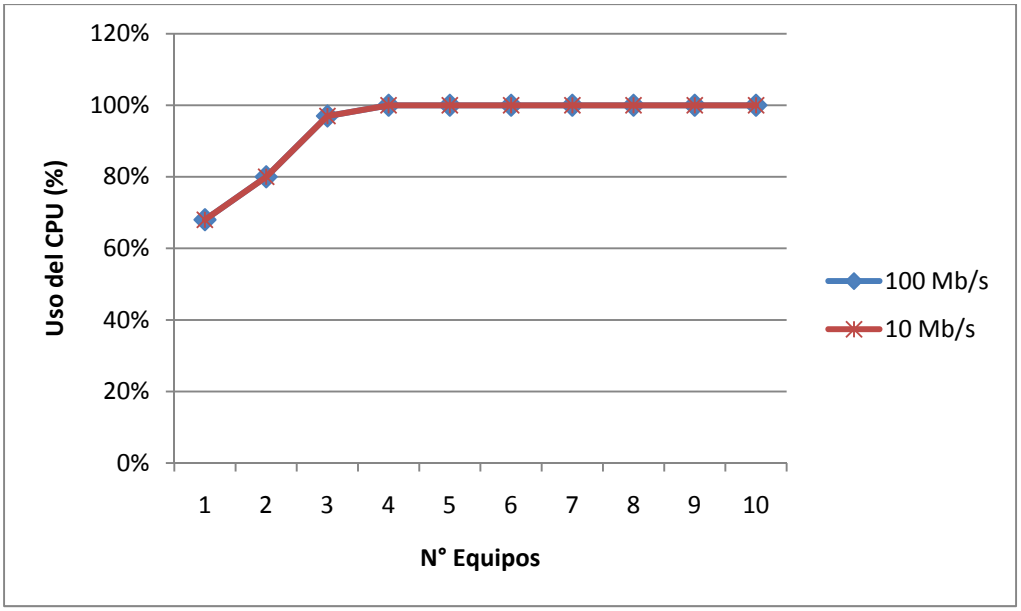

Figura 5-3: Porcentaje de Utilización del CPU v/s el Número de Equipos efectuando Peticiones al Servidor

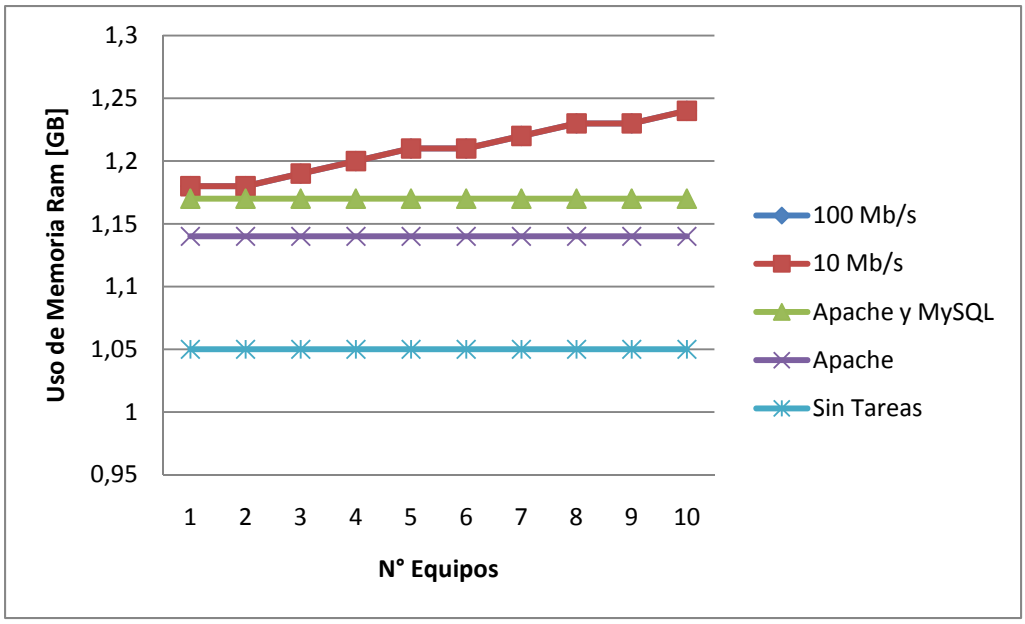

Figura 5-4: Memoria RAM utilizada v/s el Número de Equipos efectuando Peticiones al Servidor

Información fuente de la figura 5-2, figura 5-3 y figura 5-4 se encuentra en el Anexo E: Resultados de Pruebas.

Tras analizar los resultados anteriores, se concluye que la respuesta del servidor no es favorable. Jacob Nielsen, llamado “the guru of Web page usability” (el gurú de la usabilidad de páginas web) por el periódico The New York Times, escribió un artículo [19] en el cual reseña que una demora de 10 [s] en el tiempo de respuesta generalmente hará que el usuario deje el sitio y pierda interés. Como las curvas de la figura 5-2 son muy similares a una recta, se puede extrapolar el número de clientes simultáneos necesarios para lograr un tiempo de respuesta de 10 [s]. Ese número es de 17 usuarios, lo cual indica que el sistema no es lo suficientemente escalable como para ser implementado a nivel nacional, quizás inclusive a nivel regional.

Hay que recordar que este servicio se llama sólo al inicio de la aplicación, una vez recibida la respuesta no se llamará a ningún servicio salvo cuando el usuario concluya su pedido. Además, las aplicaciones que invoquen el servicio getItems no afectarán en ninguna medida a los clientes que ya están haciendo uso de la aplicación, éstos, son los que invocaron con antelación dicho servicio. Cabe destacar que 17 es el número de usuarios

concurrentes iniciando la aplicación, pero que el número de usuarios haciendo uso de ella puede ser mucho mayor.

El equipo respondió favorablemente en cuanto a lo que memoria RAM se refiere. Ésta depende linealmente de la cantidad de servicios en ejecución y crece en alrededor de 5 [MB] por cada llamado, es decir por instanciación del servicio. Como el servidor cuenta con 3 [GB] el número de instancias simultáneas puede ser muy elevado, lo cual ayudaría a la escalabilidad del sistema. Sin embargo, el desempeño del CPU no fue satisfactorio, se notó que luego de 4 clientes simultáneos, el uso del núcleo llegaba a 100 %. Esta puede ser una de las razones que están afectando la escalabilidad del sistema.

El equipo en el cual se realizaron las pruebas no es uno que esté diseñado para hacer las labores de un servidor web. Los servidores son creados específicamente para trabajar, respondiendo peticiones de servicios, construyendo complicados objetos para el cliente y realizando llamados a las bases de datos. Si se realizasen las pruebas en uno de estos equipos, el número de clientes que podrían realizar llamados simultáneos a servicios complejos sería mayor, sin embargo, los resultados del test de rendimiento no cambiarían tan drásticamente como para poder implementar el sistema a nivel nacional.

Tras ver las tablas que se encuentran en el Anexo E: Resultados de Pruebas, se concluye también, que parte importante del problema de escalabilidad es el Zend Framework. Para mayores detalles ver Anexo G: Estrategia para Mejorar el Desempeño del Servidor.

Pero no todas las conclusiones son negativas. Las curvas de respuesta obtenidas en la figura 5-2, tienen un alto grado de parecido entre ellas. Es claro que los tiempos de respuesta entre las pruebas realizadas a 100 [Mbps] y a 10 [Mbps] son casi idénticos, de lo cual se infiere, que el ancho de banda no tiene repercusión en el desempeño del servidor. Lo anterior se debe a que los paquetes enviados desde el servidor al cliente, tenían un tamaño cercano a 10 [KB]. Considerando que el servidor cuenta con una conexión del orden de 10 [Mbps], se pueden responder a un muy elevado número de clientes simultáneamente. Esta conclusión es importante ya que, en caso de que el sistema solucione su problema de escalabilidad, no será necesario invertir en una conexión a Internet de alta

velocidad para el servidor, lo cual ahorrará una inversión inicial, al momento de su implementación.

Conclusiones

Tras haber finalizado con el desarrollo de este proyecto, se concluye que la mayoría de los objetivos planteados en el inicio fueron cumplidos cabalmente. Sin embargo, uno de ellos no fue cumplido, el cual tiene una tremenda relevancia al momento de elegir cómo implementar este servicio.

Se logró construir aplicaciones, tanto para el lado del cliente como para el del administrador, que poseen interfaces intuitivas y que minimizan los posibles errores cometidos por los usuarios. Al estar ambas aplicaciones diseñadas pensando en una máquina de estados, el número de acciones posibles de los usuarios en cada módulo se vio limitado, lo cual se tradujo en una reducción importante de los errores que pudiesen realizar éstos. Se debe destacar que el diseño de la aplicación cliente permite que ésta pueda ser utilizada por un usuario promedio sin capacitación alguna.

Un objetivo importante que fue logrado es el amplio abanico de empresas al que se puede enfocar el sistema. Este producto es aplicable no sólo a entidades comerciales dedicadas al área gastronómica, sino que también a cualquier empresa que desee implementar un servicio de reparto a domicilio, prescindiendo en caso de desearlo, de la aplicación cliente.

La modularidad y flexibilidad que fueron conseguidas en este trabajo son dos de las virtudes más importantes que posee el sistema desarrollado. SIMDOC fue creado pensando en un sistema modular, lo cual se traduce en una absoluta flexibilidad al momento de crear nuevos módulos que agreguen mayor funcionalidad al sistema. Estos nuevos módulos serán absolutamente independientes de los ya construidos y podrán ser agregados con mucha facilidad a la aplicación administrador, también a la del cliente en caso que corresponda. Esta flexibilidad no es sólo en cuanto a agregar diferentes módulos sino que también al momento de diferenciar los perfiles de los servicios, es decir, ofrecer diferentes módulos en cada uno. Algunos perfiles contarán sólo con los módulos básicos, mientras que otros podrán contar con un mayor número de módulos que presenten aún más funcionalidades para las empresas.

El objetivo que no fue logrado es la escalabilidad del sistema. Los resultados arrojados por las pruebas de rendimiento, niegan tajantemente la posibilidad de implementar el proyecto a nivel nacional, quizás inclusive a nivel regional. Con un bajo número de usuarios concurrentes el sistema se satura, introduciendo importantes retardos, por lo que sería imposible intentar albergar un alto número de empresas. Si bien las pruebas fueron realizadas en un equipo no idóneo y que los resultados obtenidos cambiarían al realizarse en un servidor, también se debe mencionar que el escenario de pruebas utilizado no cumple con los estándares actuales, es decir, el modelo de pruebas debiese reflejar con mayor exhaustividad el escenario real en el cual se desempeñará el sistema. Ese cambio en el modelo de pruebas reducirá el desempeño de SIMDOC de manera significativa, contraponiéndose al aumento en el desempeño que entregará el realizar las pruebas en un servidor real.

El Flex Framework cumplió con sus objetivos, ya que ayudó a crear aplicaciones ricas en contenido, amigables para cualquier usuario y a disminuir los posibles errores que puede cometer un usuario a la hora de utilizar la aplicación. Sin embargo, el Zend Framework no cumplió completamente con los resultados esperados. Si bien fue útil al momento de ayudar a desarrollar un sistema independiente del motor de base de datos y del formato usado para el intercambio de datos entre las aplicaciones y el servidor, fue el uso de este framework lo que impidió la escalabilidad del sistema. Al estudiar con mayor profundidad los resultados obtenidos en la fase de testing, se nota inmediatamente que la mayor parte del tiempo empleado para que retornen los resultados al cliente transcurre en la invocación de los métodos del Zend Framework. Realizar sólo una llamada al servidor, del mismo tipo que las realizadas en la fase de testing, toma alrededor de 700 [ms], sin embargo, al desactivar el framework y cambiar el modo de trabajo se logran resultados que oscilan cerca de los 70 [ms]. En el Anexo G: Estrategia para Mejorar el Desempeño del Servidor se plantea el método utilizado para lograr estos resultados. Es necesario destacar, que estas nuevas medidas de desempeño significan una notable mejora en lo relacionado a la escalabilidad del sistema, lo que implica un aumento importante en el número de empresas que podrá albergar el sistema.

Por todo lo que se ha señalado anteriormente, se concluye que es posible usar SIMDOC para generar un negocio que poseerá un atractivo comercial tanto para quien lo implementa, como para quien contratará el servicio. Sin embargo, no se podrá implementar a nivel nacional como se había buscado, sino que deberá ofrecer sus servicios a un número limitado de empresas.

Algunos puntos de interés que se deberían considerarse en las futuras versiones posteriores de este sistema:

- Cadenas de restaurantes: En esta primera etapa, no se consideraron a las empresas que poseen múltiples locales, obligando a las interesadas a contratar un servicio por cada local, siendo cada uno de ellos completamente independiente del otro. Debe examinarse la posibilidad de optimizar el sistema para ser capaz de alojar este tipo de empresas. Será necesario crear un nuevo perfil de servicio, en el cual se cuente con una sola aplicación cliente para todos los locales. Se deberá entregar una aplicación administrador central que se dedique sólo a administrar la aplicación cliente y los productos que en ella se despliegan. Además deberá proporcionarse aplicaciones administrador a cada local, que sólo cuenten con módulos que les permitan administrar su servicio de reparto.
- Nivel mundial: Si bien el proyecto fue pensado para nivel nacional, no se ven factores que impidan la implementación de este servicio a nivel mundial, modificando las aplicaciones de manera que detecten en qué país están siendo utilizadas y así cambiar el lenguaje de las mismas permitiendo, de esta manera, un uso global de SIMDOC.
- Múltiples servidores: Se desea también, plantear la idea de implementar el sistema en múltiples servidores, para que sea capaz de soportar un muy alto número de clientes simultáneos. Este punto, es de vital importancia si se piensa desarrollar el punto anterior.
- Interfaz WYSIWYG: Para una mayor simplicidad a la hora de utilizar la aplicación administrador, se podría implementar un sistema de edición del tipo WYSIWYG. Lo anterior ayudaría a usuarios que tienen conocimientos computacionales mínimos, a utilizar la aplicación casi sin instrucciones.

- Personalización del cliente: Si bien la aplicación del cliente tiene un grado de personalización, la aplicación cliente deberá ser completamente editable, desde las fuentes hasta los “thumbnails” diseñados para mostrar los platos. Esto, con el fin de que cada empresa pueda tener su propia imagen corporativa, completamente distinta del resto de las empresas que también han contratado el servicio.

Anexos

Anexo A: Presupuesto Para un Sitio Web

A continuación se explican las metodologías que se utilizaron para obtener los costos necesarios para desarrollar una página web.

Servicio de Webhosting

Para obtener una estimación de este valor en el mercado, se fijaron una serie de requerimientos que se consideraron necesarios para el desarrollo de un sitio con comercio electrónico. Luego se procedió a buscar empresas que cumpliesen con ellos y así obtener los presupuestos respectivos.

Requerimientos: Espacio de almacenamiento mayor a 1 [Gb]. Número de cuentas de correo mayor a 50. Tráfico ilimitado o mayor a 200 [Gb] mensuales. Con más de una base de datos.

En la tabla anexo A-1 se detallan las empresas que cumplieron con los requisitos y el valor de dichos planes (precios sin IVA):

Tabla Anexo A-1: Cotizaciones de planes servicio de webhosting

Empresa	Plan	Precio [anual]	Sitio web
PowerHost	Hosting Titanium	$23.000	http://www.powerhost.cl/hosting
Dattatec	Plan Inicio	$21.900	http://dattatec.com/site/sp/chile/web-hosting#Detalles
TChile	Económico	$23.000	http://www.tchile.com/virtual.php
Planeta Hosting	Plan Estandar	$29.900	http://www.planetahosting.cl/estandar.html

Se estimó que el valor para un servicio de webhosting que cumpla con los requisitos estipulados anteriormente es de $23.000 anuales.

Diseño de un Sitio Web con Comercio Electrónico

En este ítem se buscó un sitio web que implementase comercio electrónico y que cumpliese sólo con las necesidades básicas. Una vez encontrado el sitio, se procedió a solicitar a diferentes empresas que se dedican al rubro, presupuestos para desarrollar una página con las mismas características que la que se les enseñó.

Página utilizada como muestra: http://www.electrotemporada.cl/index.php

En la tabla anexo A-2 se entregan los presupuestos obtenidos por las diferentes empresas a las cuales se les solicitó (Sin IVA):

Tabla Anexo A-2: Cotizaciones para un sitio web con comercio electrónico

Empresa	Presupuesto	Página web
Ilógica	$1.600.000	http://www.ilogica.cl
W3 Estudios	$350.000	http://www.w3-studios.com/
Todo Soporte	$450.000- $500.000	http://www.todosoporte.cl/
Pymes en Chile	$370.000	http://www.pymesenchile.cl/
Su Sitio Web	$419.000	http://www.susitioweb.cl/
CMW Chile	$350.000	http://www.cmwchile.com/

Ilógica fue considerado por la relevancia de las empresas que se encuentran en su portafolio, a diferencia del resto de las empresas mencionadas. Entre ellas se encuentran Sopraval, las universidades Federíco Santa María y Adolfo Ibáñez, Esval y Ciderval.

Los precios oscilan en valores cercanos a los $400.000 (sin IVA), por lo tanto será este el valor estimado que se tomará para calcular el total necesario para desarrollar un sitio web.

Anexo B: Casos de Uso

Caso de uso: Añadir/Editar Categoría.

Actores: Administrador

Propósito: Agregar una categoría a un restaurante.

Precondiciones: Debe estar logeado.

Postcondiciones: Se agregará o actualizará, una categoría del restaurante actor.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **agregar categoría** o en **editar categoría**.
2. El administrador advertirá una ventana de edición de categoría con los campos asociados a ésta y una lista de los platos asociados a ella.
3. El usuario podrá editar cada valor, si intenta modificar o agregar un plato, se trasladará al caso de uso “Añadir/Editar Plato”.
4. Al presionar **aceptar** se ejecutará un script en el servidor, que guardará los cambios en la base de datos.

Tipo: Primario

Caso de uso: Borrar Categoría.

Actores: Administrador

Propósito: Borrar una categoría de un restaurante.

Precondiciones: Debe estar logeado.

Postcondiciones: Se eliminará una categoría del restaurante actor, con la totalidad de sus platos.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **borrar categoría**.
2. El servidor revisará si los platos se encuentran en algún pedido.
3. De ser así, el plato no se borrará de la base de datos, solo se cambiará su estado a eliminado y se desasociará, de la categoría en cuestión.
4. La categoría se eliminará de la base de datos.

Tipo: Primario

Caso de uso: Añadir/Editar Plato.

Actores: Administrador

Propósito: Agregar un plato a la categoría asociada a un restaurante.

Precondiciones: Debe estar logeado y estar dentro de la sección de edición de categoría.

Postcondiciones: La categoría asociada, agregará o actualizará un plato.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **agregar plato** o en **editar plato**.
2. Si hizo click en **agregar plato**, se creará un registro nula en la base de datos y se recuperará la entrada para poder contar con el id del plato.
3. El usuario distinguirá una ventana de edición del plato con todos los campos, con excepción del id y la versión de éste.
4. El usuario modificará a voluntad los valores y podrá hacerlo hasta, que presione aceptar.
5. Al apretar aceptar, se enviará la información al servidor donde se cambiará el registro coincidente con el id del plato.

Tipo: Primario

Caso de uso: Borrar Plato.

Actores: Administrador.

Propósito: Borrar un plato de una categoría.

Precondiciones: Debe estar logeado y estar dentro de una sección de edición de categoría.

Postcondiciones: Se eliminará el plato.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **borrar plato**.
2. El servidor cambiará el estado del plato a eliminado.

Tipo: Primario.

Caso de uso: Añadir/Editar Repartidor.

Actores: Administrador.

Propósito: Agrega o modifica una cuenta de repartidor.

Precondiciones: Debe estar logeado.

Postcondiciones: Se agregan o modifican los datos del repartidor en la base de datos.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **agregar repartidor** o en **editar repartidor**.
2. Si hizo click en **agregar repartidor**, se creará uno registro nulo en la base de datos y se recuperará la entrada para poder contar con el id del repartidor.
3. El usuario observará una ventana con la información del repartidor y podrá modificarla.
4. Al apretar aceptar, se guardará la información en la base de datos.

Tipo: Primario.

Caso de uso: Borrar Repartidor.

Actores: Administrador.

Propósito: Borrar un plato de una categoría.

Precondiciones: Debe estar logeado y estar dentro de una sección de edición de categoría.

Postcondiciones: Se eliminará el repartidor.

Curso Básico de Acción:

1. El restaurante hará click en el botón de **borrar repartidor**.
2. El servidor revisará si el repartidor, tiene pedidos pendientes.
3. De ser así, se le informará al restaurante que no puede ser borrado hasta que haya finalizado el pedido.
4. En caso contrario se borrará la cuenta del repartidor.

Tipo: Primario.

Anexo C: Ciclo de Vida de una Llamada Remota

En la figura anexo c-1 se aprecia un diagrama de flujo de la información en el uso de llamadas remotas con AMF.

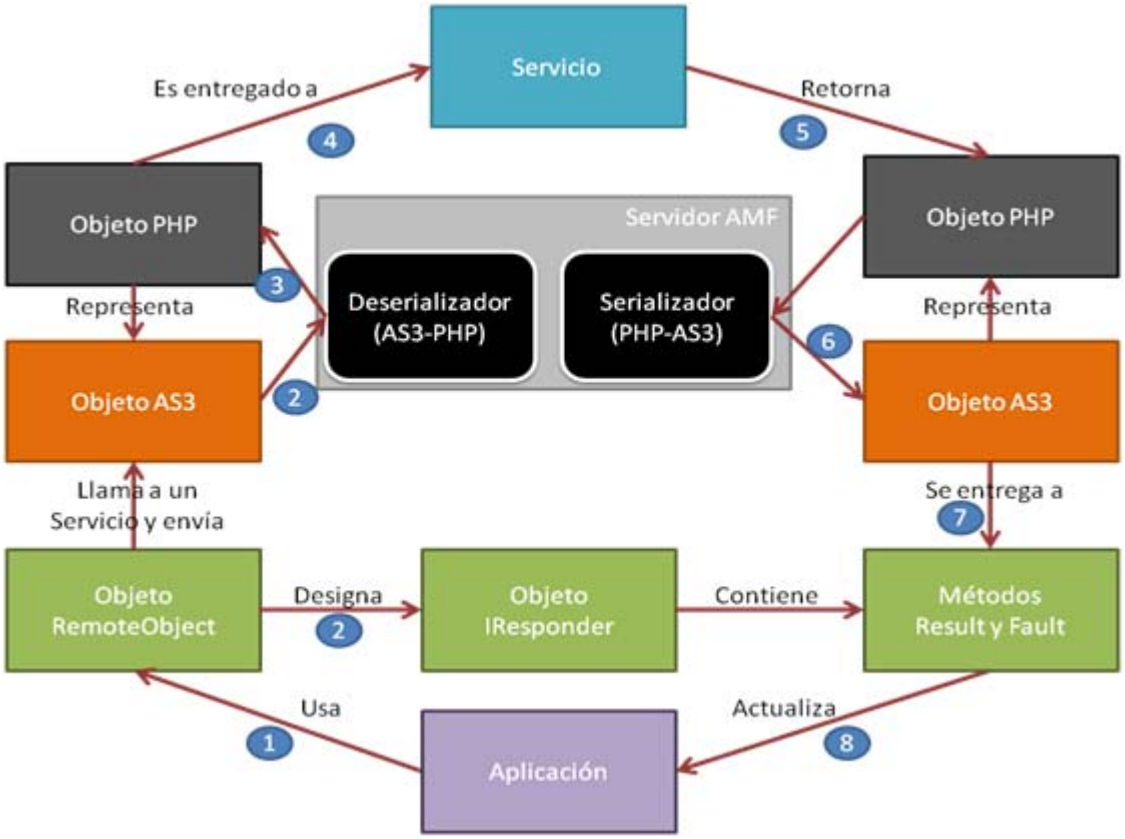

Figura Anexo C-1: Diagrama del Flujo de la Información en el Uso de Servicios Remotos

1. Primero la aplicación instancia un objeto **RemoteObject** y le solicita que llame a un servicio y le entrega los parámetros.
2. El objeto **RemoteObject** efectúa la solicitud a la dirección URL del servidor AMF y envía los parámetros tal cual los recibió. Hecho esto, designa un objeto **IResponder** el cual tiene dos métodos, **result** y **fault**.
3. La solicitud y los parámetros, son deserializados en el Servidor y transformados en objetos PHP.
4. El servidor llama al servicio descrito dentro de la solicitud y le entrega estos objetos como parámetros.

5. El servicio se ejecuta y retorna un objeto PHP.
6. Este objeto PHP es serializado por el servidor AMF y es retornado al cliente.
7. En caso de haber ocurrido algún problema o dificultad en todo el flujo descrito, el objeto **IResponder** designado ejecutará el método **fault**, en caso contrario se ejecutará el método **result** con el objeto recibido como parámetro.
8. El método **result** actualiza el modelo de la aplicación.

Anexo D: Sombrero Blanco contra Sombrero Negro

Las técnicas de SEO (Search Engine Optimization) se clasifican en dos grandes categorías: las técnicas que los buscadores recomiendan como parte de un buen diseño y las técnicas que los buscadores desaprueban, con el fin de minimizar la indexación errónea. Algunos clasifican estos métodos como SEO de sombrero blanco y SEO de sombrero negro ("White hat SEO and Black hat SEO") respectivamente. Los sombreros blancos tienden a producir resultados que perduran en el tiempo, mientras que los sombreros negros anticipan que sus sitios, con el tiempo serán penalizados y eventualmente prohibidos, una vez que los motores de búsqueda descubran lo que están haciendo.

Una táctica de SEO se considera de sombrero blanco si se ajusta a las directrices de los buscadores y no implica ningún engaño. SEO de sombrero blanco no se trata sólo acerca de las directrices, sino que trata de garantizar que el contenido que un motor de búsqueda indexa y clasifica, es el mismo contenido que ofrecerá al usuario.

El sombrero blanco aconseja tener como primera prioridad la creación de contenido para los usuarios, no para los motores de búsqueda. Luego se enfoca en facilitar el acceso al contenido para los robots. La optimización de sombrero blanco es, en muchos aspectos, similar al desarrollo web que promueve la accesibilidad, aunque los dos no son idénticos.

El uso del sombrero blanco es un eficaz medio de marketing, que realiza esfuerzos para ofrecer contenidos de calidad a un público objetivo. Los medios tradicionales de marketing han permitido esto a través de la transparencia y la exposición. Un algoritmo de un motor de búsqueda, que tiene esto en cuenta, es el PageRank de Google.

El sombrero negro intenta mejorar el ranking en formas que no son aprobadas por los motores de búsqueda, y por lo general son un engaño. Una de las técnicas que el sombrero negro utiliza, es el contenido oculto, ya sea usando un color de texto similar al del fondo, en un tag <DIV> invisible o colocado fuera de la pantalla. Otro método es mostrar una página diferente si la solicita un humano o un motor de búsqueda, técnica conocida como encubrimiento.

Los motores de búsqueda pueden penalizar a los sitios que son descubiertos utilizando métodos de sombrero negro, ya sea mediante la reducción de su clasificación o eliminación de sus anuncios de sus bases de datos por completo. Estas sanciones pueden ser aplicadas de forma automática por los algoritmos de los buscadores, o por una revisión manual de los sitios. Un ejemplo famoso fue en febrero de 2006 la eliminación de los dos sitios: BMW Germany y Ricoh Germany, debido al uso de prácticas engañosas. Ambas empresas, sin embargo, se disculparon rápidamente y repararon las páginas infractoras. Finalmente fueron restaurados a la lista de Google.

Anexo E: Resultados de Pruebas

A continuación, se detallarán los resultados obtenidos tras realizar las pruebas de rendimiento, mencionadas en el capítulo 5:

Tabla Anexo E-3: Rendimiento del servidor en escenario de 100 Mb/s

# Clientes	Total [ms]	En/Re [ms]	Red[ms]	Server [ms]	"Service Call" [ms]	CPU [%]	RAM [GB]
1	735	727	154	573	493	68	1,18
2	1093	1086	162	924	831	80	1,18
3	1874	1868	166	1702	1562	97	1,19
4	2765	2764	207	2557	2349	100	1,2
5	3404	3398	168	3229	3091	100	1,21
6	3777	3768	169	3599	3459	100	1,21
7	4132	4145	161	3989	3803	100	1,22
8	4687	4701	158	4352	4201	100	1,23
9	5209	5191	170	4894	4715	100	1,23
10	6120	6101	165	5809	5673	100	1,24

Tabla Anexo E-4: Rendimiento del servidor en escenario de 10 Mb/s

# Clientes	Total [ms]	En/Re [ms]	Red[ms]	Server [ms]	"Service Call" [ms]	CPU [%]	RAM [GB]
1	699	688	157	529	455	68	1,18
2	1105	1009	155	861	799	80	1,18
3	1910	1907	160	1752	1629	97	1,19
4	2703	2682	193	2485	2439	100	1,2
5	3359	3343	192	3122	2998	100	1,21
6	3692	3615	172	3400	3321	100	1,21
7	4221	4206	169	4045	3922	100	1,22
8	4731	4712	168	4544	4252	100	1,23
9	5149	5122	158	4966	4841	100	1,23
10	6088	6045	159	5911	5832	100	1,24

Anexo F: Estrategia de Prueba de Rendimiento para SIMDOC

En el presente anexo, se muestra una estrategia diseñada específicamente para medir la respuesta del sistema frente a una alta cantidad de usuarios concurrentes. Se explicarán, cada uno de los detalles del escenario seleccionado y lo necesario para la implementación del mismo. Además, se incluyen en esta sección los fundamentos por las cuales esta estrategia no pudo ser implementada en este proyecto.

Algunos alcances

Tras una amplia investigación referente a las pruebas de rendimiento, se ha concluido que no existen estándares universales para la realización de este tipo de pruebas, tanto para la selección de los escenarios de prueba como para la elección de los criterios de aceptación del rendimiento. Cada proyecto es único y diferenciable en cuanto a su implementación y expectativas, es por ello que las únicas máximas que se encuentran al momento de realizar una estrategia de pruebas son:

- Los escenarios seleccionados deben ser los más determinantes al encontrarse el sistema en el ambiente de producción.
- Los modelos de trabajo de carga deben representar, tanto el comportamiento de cada usuario, como la carga a la que se someterá el sistema.

- Los parámetros a medir, deben ser acorde con los objetivos de la medición.

Esta estrategia de pruebas se realizó tomando en cuenta los consejos entregados por Microsoft [20], LoadStorm [21] y Websphere [22].

Acerca de la estrategia implementada

El test de rendimiento que se ejecutó en este trabajo no refleja una medición objetiva, esto se debe a que el escenario elegido no se asemeja al escenario real en el cual el servidor estará trabajando. El modelo de pruebas cumple sólo con la tercera máxima de las mencionadas anteriormente. Sin embargo, al momento de decidir cómo realizar las pruebas se optó por éste, debido a la imposibilidad de poner en funcionamiento la estrategia mencionada en este anexo.

Se deben establecer las razones, por las que no se implementó este modelo de pruebas en este proyecto:

- Los programas utilizados en esta estrategia son desarrollados por empresas como HP, Compuware e IBM. Cada uno de ellos para ser adquirido requiere de una importante inversión, la cual no se contempló al comenzar el proyecto.
- Para lograr los objetivos de dichos software es necesaria una capacitación, lo cual tomaría una cantidad de tiempo con la cual no se cuenta.
- Para desarrollar la presente prueba se debe contar con un equipo que haya sido diseñado para ejecutar las labores de un servidor web, equipo con el cual tampoco se contaba al momento de realizar las pruebas.

Es por estas razones por lo que se optó por elaborar una metodología que sí estuviese al alcance de los desarrolladores del proyecto, aún en desmedro de la objetividad del análisis.

Objetivo de la estrategia

El objetivo de esta prueba, es fijar el número máximo de usuarios concurrentes que puede tolerar el sistema, es decir, verificar si efectivamente puede ser considerado para uso masivo o no.

Selección de Escenarios de Usuarios

Los escenarios por parte del cliente son:

- Navegar por los productos
- Realizar un pedido

Mientras que los más importantes por parte del administrador son:

- Gestionar un pedido
- Modificar un pedido

Estos cuatro escenarios de usuarios deben estar presentes en el escenario de pruebas a realizar.

Elección del Modelo de Trabajo de Carga

A continuación, se detallarán tres perfiles de usuario que serán implementados como usuarios virtuales para realizar las pruebas. Cada uno de ellos detalla los pasos que seguirá ese tipo de usuario.

- **Consumidor observador**:
    1. Algunos de estos usuarios descargarán la aplicación cliente desde el servidor, otros no, porque se asumirá que un porcentaje de ellos ya tienen almacenada la aplicación en su ordenador.
    2. Se llamará al servicio getInfo en el servidor para descargar todo el catálogo. Como se obtendrá el catálogo completo, aquí terminará este perfil, ya que para navegar por los productos no se necesita realizar ningún otro llamado.
    3. Se termina con el usuario virtual.
- **Consumidor comprador**:
    1. No se descargará la aplicación cliente desde el servidor, se asume que este tipo de usuario ya ha visitado la página con anterioridad y por lo tanto, tiene almacenada la aplicación en su ordenador.
    2. Se llamará al servicio getInfo en el servidor para descargar todo el catálogo.
    3. Se introducirá una pausa variable, con el fin de reflejar la experiencia que tiene un usuario real al momento de elegir sus productos.
    4. Se realiza un llamado al servicio realizarPedido en el servidor, para que en este último se almacene toda la información pertinente al pedido.
    5. Se termina con el usuario virtual.

- **Administrador 1**:
    1. No se descargará la aplicación administrador desde el servidor, por las mismas razones que el usuario anterior.
    2. Se llamará al servicio getInfo en el servidor, para descargar toda la información pertinente a esta aplicación.
    3. Se llamará al servicio getPedidos, para obtener el listado de nuevos pedidos.
    4. Pausa variable que reflejará el tiempo de procesamiento en la aplicación administrador y el tiempo de análisis del usuario.
    5. Se realiza un llamado al servicio estadoPedido en el servidor, para cambiar el estado de pedido.
    6. Pausa corta de tiempo.
    7. Se realiza un llamado al servicio asignarRepartidor en el servidor.
    8. Pausa corta de tiempo.
    9. Se realiza un llamado al servicio estadoPedido en el servidor, para cambiar el estado de pedido.
    10. Se vuelve al paso 2.
- **Administrador 2**:
    1. No se descargará la aplicación administrador desde el servidor, por las mismas razones ya aludidas.
    2. Se llamará al servicio getInfo en el servidor, para descargar toda la información pertinente a esta aplicación.
    3. Se llamará al servicio getPedidos, para obtener el listado de nuevos pedidos.
    4. Pausa variable que reflejará el tiempo que le toma al usuario modificar un pedido y el tiempo de análisis del usuario.
    5. Se realiza un llamado al servicio estadoPedido en el servidor, para cambiar el estado de pedido.
    6. Pausa corta de tiempo.
    7. Se realiza un llamado al servicio asignarRepartidor, en el servidor.
    8. Pausa corta de tiempo.
    9. Se realiza un llamado al servicio estadoPedido en el servidor, para cambiar el estado de pedido.
    10. Se vuelve al paso 2.

Las proporciones en la que aparecerán estos usuarios será de cada 100 usuarios, 82 serán Consumidor observador (entre ambas variantes), 8 serán Consumidor comprador, 8 del tipo Administrador 1 y 2 del tipo Administrador 2.

La estimación anterior refleja una tendencia parecida al ejemplo citado en el artículo desarrollado por LoadStorm [21]. Sin embargo, como cada uno de estos perfiles debe estar

en la proporción correcta, es aconsejable realizar un estudio de mercado acabado relacionado con “ecommerce” para así poder elegir proporciones más apropiadas.

Niveles de Carga Objetivos

Los niveles de carga comenzarán con un total de 30 usuarios, luego se irán aumentando paulatinamente. Se añadirá un total de 30 usuarios virtuales, guardando las proporciones que fueron estipuladas en el punto anterior.

Métricas a Evaluar

Como el objetivo lo indica, se busca evaluar si esta aplicación es escalable. Para determinar lo anterior se deben tomar en cuenta dos factores: el número de usuarios que puede tolerar el sistema y el tiempo de respuesta percibido por cada usuario. Estos dos elementos determinarán si es posible implementar SIMDOC a nivel nacional. Por lo tanto, se medirán los siguientes indicadores: tiempo de respuesta por cada llamada al servidor, uso de memoria RAM y de CPU en el servidor.

Escenario de Red

En el siguiente esquema se establece cuál será el escenario sobre el que se montarán las pruebas:

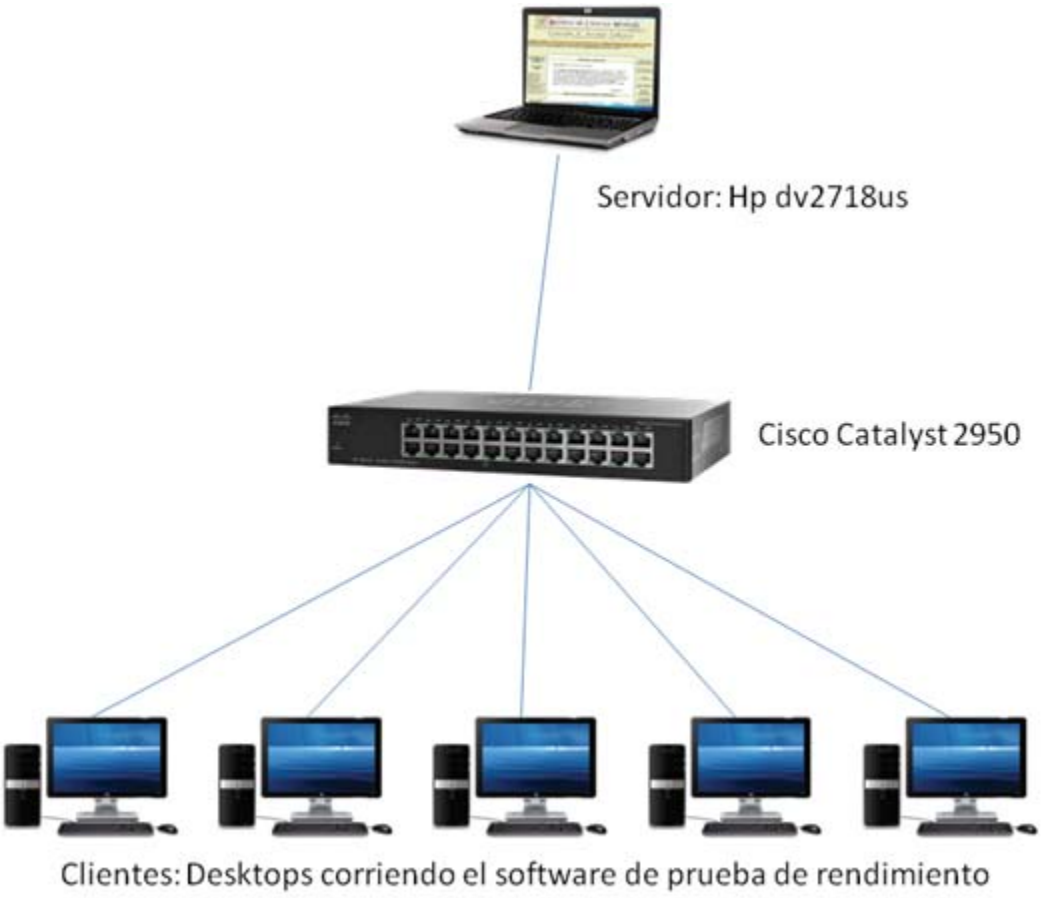

Figura Anexo F-2: Diagrama de red utilizado para realizar las pruebas

Cada equipo se ocupará de simular un número determinado de usuarios virtuales, pertenecientes a sólo un perfil de usuario, es decir, un equipo se encargará de simular los usuarios Consumidor observador variante 1, otro los usuarios Consumidor observador variante 2, así hasta completar los cinco tipos de usuarios. Cada ordenador debe contar con un software especializado en **análisis de stress/carga de un servidor web**. Algunos de los programas más reconocidos son LoadRunner de Hp/Mercury, QALoad de Compuware y Rational Performance Tester de IBM. Se correrá sólo una instancia, de cada uno de estos programas en cada equipo. Este software se encargará de realizar el procedimiento pertinente a cada perfil para el número de usuarios que está simulando. El mismo software se encargará de realizar las mediciones relacionadas el tiempo de respuesta y luego graficarlas al terminar el análisis. Estos equipos deben tener un procesador de al menos 1.5 [GHz], preferentemente multi-núcleo, además de 1 [GB] de memoria RAM.

El servidor entrega los indicadores que se mencionaron en el punto anterior, pero no los almacena, por lo que habrá que modificarlo para que lo realice y luego puedan ser analizados. Este equipo deberá contar con un procesador de al menos 2.5 [GHz] y al menos con 4 núcleos, en cuanto a memoria de RAM deberá contar con al menos 3 [GB].

En este escenario de red global existirán varios sub-escenarios en los que se variará el ancho de banda. El inicial de cada enlace será de 100 [Mbps], se irá disminuyendo en 10 [Mbps] en cada medición hasta llegar a los 10 [Mbps].

Realización del test y aplicación del criterio de aceptación de rendimiento

Para cada sub-escenario, se iniciará con un número de 30 usuarios totales. Se irá aumentando dicho número en el factor mencionado previamente hasta llegar a un tiempo de respuesta de 10 [s]. Una vez logrado ese tiempo se continuará con el siguiente sub-escenario.

Realizados los 10 sub-escenarios, se graficarán todos los factores obtenidos en cada medición. Según el artículo de Jacob Nielsen [19] existen 3 tiempos de respuesta claves en las aplicaciones web:

- 0.1 [s] para que el usuario sienta que tiene absoluto control de la aplicación.
- 1 [s] para que el flujo de pensamiento del usuario no sea interrumpido y éste siga sintiendo que mantiene control de la aplicación.
- 10 [s] mantiene la atención del usuario, pero éste siente que está a expensas del equipo y que no tiene control sobre la aplicación.

Sin duda que para lograr una aplicación satisfactoria para el cliente, el rango debe encontrarse entre los últimos dos tiempos, ya que para una aplicación masiva el primero es imposible de lograr. Se decidió fijar el límite de usuarios máximos tolerados por el sistema para un tiempo de respuesta de 4.5 [s], de esta manera el tiempo es suficientemente bajo como para que el usuario, se siga manteniendo cómodo. En cada sub-escenario se deberá encontrar ese tiempo y fijar para ese número de usuarios, el límite.

Anexo G: Estrategia para Mejorar el Desempeño del Servidor

Los resultados obtenidos en las pruebas hechas son bastante desalentadores, el tiempo de CPU usado en la recopilación de datos supera por mucho el aceptable.

Al momento de identificar el problema lo primero que salta a la mente es la codificación a AMF. Al revisar los tiempos, para el caso de una solicitud aislada se obtiene que la codificación tarda alrededor de 60 [ms] mientras que el uso de “Zend” para obtener la información demora aproximadamente 600 [ms]. Viendo esto, se omitió el uso de “Zend Framework” para la conexión con la base de datos y se insertó la “query” SQL directamente en el código. Los cambios no se hicieron esperar, la recopilación de datos se redujo a 25 [ms] y la codificación se mantuvo en 60 [ms].

Aun 100 [ms], considerando la decodificación de la llamada remota, sigue siendo excesivos para una solicitud aislada. Hay que hacer la diferencia, entre los servicios que requieren necesariamente de una base de datos y los servicios que lo hacen parcialmente. Para recopilar información que perdurará sin variar en el tiempo no es necesario acceder a la base de datos, basta con un archivo que contenga la respuesta ya generada. Usando este concepto, se diseñó un sistema que genera un archivo con la respuesta ya codificada en AMF. De este modo, si el archivo de caché no existe, el servicio se ejecuta de forma normal, y antes de enviar la respuesta, esta se guarda en disco. El caso anterior tarda 100 [ms], mientras que en la situación donde el archivo existiese, el servicio retorna directamente su contenido, tardando 20 [ms] en responder, de esta manera el sistema ahorra 80 [ms] ya que no necesita recopilar la información ni codificarla.

El archivo de cache, una vez creado, se mantiene en disco hasta que el administrador del restaurant realiza un cambio en la base de datos. En ese momento, se borra el archivo de cache y no se volverá a generar hasta que un cliente solicite el servicio nuevamente. De esta forma el servicio tardará 100 [ms], para el primer usuario que navegue por la aplicación cliente desde el último cambio realizado en la base de datos, mientras que para todo el resto de usuarios tardará solo 20 [ms].

En la figura anexo g-3 y la figura anexo g-4 se ven los diagramas de flujo resultantes para esta propuesta.

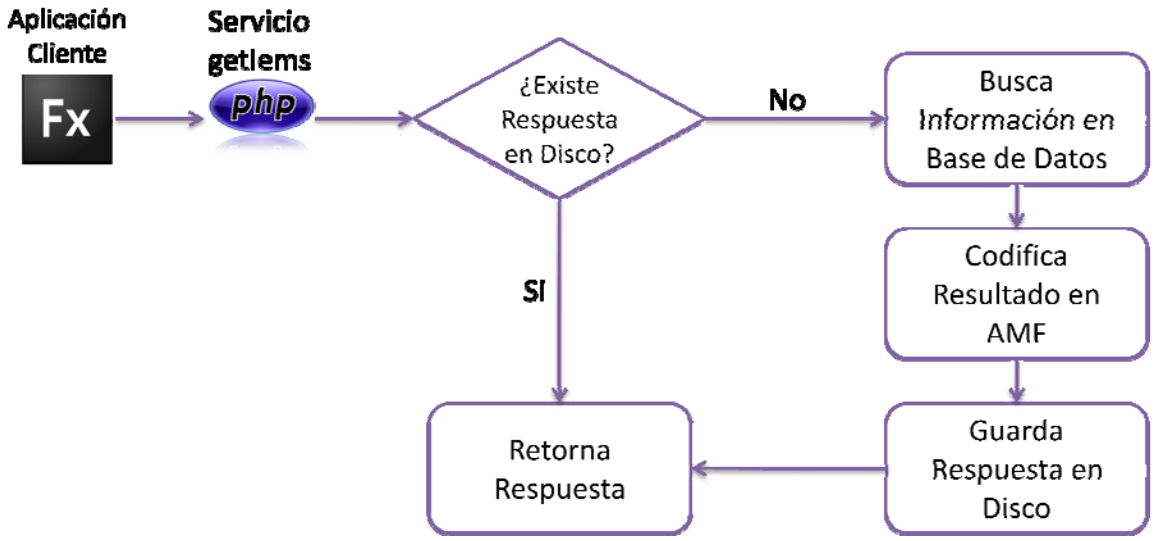

Figura Anexo G-3: Diagrama de flujo de solución planteada para obtener información

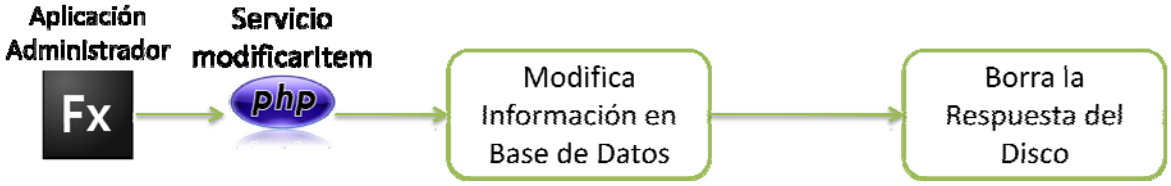

Figura Anexo G-4: Diagrama de flujo de solución planteada para modificar información

Referencias

[1] World Internet Usage Statistics News and World Population Stats [en línea]
http://www.internetworldstats.com/stats.htm
[consulta: Agosto 2010]

[2] Google revela el uso que dan a internet las pymes de América Latina [en línea] 23 Mayo 2010
http://fanultra.posterous.com/google-revela-el-uso-que-dan-a-internet-las-p
[consulta: Septiembre 2010]

[3] NIC Chile [en línea] 1 julio 2010
http://www.nic.cl/aranceles.html
[consulta: Agosto 2010]

[4] Consultora Innoves Ltda. [en línea]
http://www.innoves.cl/
[consulta: Julio 2010]

[5] Methodo Ltda. [en línea]
http://www.methodo.cl/
[consulta: Julio 2010]

[6] Axoft Chile S.A. [en línea]
http://www.resto.cl/
[consulta: Julio 2010]

[7] Laudus S.A. [en línea]
http://www.laudus.cl/
[consulta: Julio 2010]

[8] ARINA Teemu, OY Dicole, Web 2.0 Business Models [en línea], Febrero 2008.
http://www.slideshare.net/infe/web-20-business-models-270855
[consulta: Septiembre 2010]

[9] RAPPA Michael, Business Models on the Web [en línea]
http://digitalenterprise.org/models/models.html
[consulta: Septiembre 2010]

[10] O'REILLY Tim, What Is Web 2.0 [en línea]
http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=2
[consulta: Septiembre 2010]

[11] CRUD - Wikipedia, la enciclopedia libre [en línea]
http://es.wikipedia.org/wiki/CRUD
[consulta: Febrero 2011]

[12] ACID - Wikipedia, la enciclopedia libre [en línea]
http://es.wikipedia.org/wiki/ACID
[consulta: Febrero 2011]

[13] DERAEDT David, Flex Architecture Fundamentals [en línea] Febrero 2008.
http://www.dehats.com/drupal/?q=node/32
[consulta: Julio 2010]

[14] Medieros Miller, SEO, Truths and Miths [en línea] Septiembre 2009
https://docs.google.com/present/view?id=dd2tz3mw_62fsndjtcg
[consulta: Septiembre 2010]

[15] Medieros Miller, SEO – A Quick Overview [en línea] Septiembre 2009
https://docs.google.com/present/view?id=dd2tz3mw_71ccsprmcz
[consulta: Septiembre 2010]

[16] Medieros Miller, SEO for Flash Websites [en línea] Septiembre 2009
https://docs.google.com/present/view?id=dd2tz3mw_84fsdftvw8
[consulta: Septiembre 2010]

[17] BEANSTALK, White-Hat SEO Tactics [en línea]
http://www.beanstalk-inc.com/tactics/white-hat.htm
[consulta: Julio 2010]

[18] GOOGLE, Optimización para motores de búsqueda Guía de Google para Principiantes [en línea] 21 Noviembre 2008
http://www.google.es/webmasters/docs/guia_optimizacion_motores_busqueda.pdf
[consulta: Julio 2010]

[19] Website Response Times, Jakob Nielsen [en línea] 21 Junio 2010
http://www.useit.com/alertbox/response-times.html
[consulta: Diciembre 2010]

[20] Chapter 17 – Load-Testing Web Applications, Microsoft [en línea] Septiembre 2007
http://msdn.microsoft.com/en-us/library/bb924372.aspx
[consulta: Diciembre 2010]

[21] Realistic Load Testing Scenarios, Load Strom [en línea]
http://loadstorm.com/2011/realistic-load-testing-scenarios-part-1
http://loadstorm.com/2011/realistic-load-testing-scenarios-part-2
[consulta: Enero 2011]

[22] Performance Testing Web Services, Websphere Journal [en línea]
http://websphere.sys-con.com/node/46513
[consulta: Enero 2011]