Universidad de las Ciencias Informáticas
Facultad 3

“Arquitectura para el Sistema de Administración de Relaciones con el Cliente”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Ernesto Mató Roque
Ariel Ramírez Ferriol

Tutores: Ing. Pedro Manuel Alás Verdecia
Ing. Omar Antonio Díaz Peña

La Habana, 2012.

“Año 54 de la Revolución”

DECLARACION DE AUTORIA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Informatización de Gestión de Entidades de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2012.

Ernesto Mató Roque

_____

Autor

Ariel Ramírez Ferriol

_____

Autor

Ing. Pedro Manuel Alás Verdecia

_____

Tutor

Ing. Omar Antonio Díaz Peña

_____

Tutor

"Seamos realistas y hagamos lo imposible."

Agradecimientos

Quiero agradecer en primer lugar a mis padres (Margarita y Rodolfo), mis hermanos (Niurka, Yaleidis y Alejandro), mis abuelos (Vidalina, Olegario, Elsi Ester y Ciro Arnoldo) que aunque algunos de ellos no estén presentes físicamente, siempre me anhelaron un buen futuro, a mis tíos y primos (Aracelis, Niuvis, Yailin, Jorge y Ernesto) que fueron mis segundos padres y hermanos mientras estuve estudiando en la universidad, a las siempre inquietas primitas e inquietos primitos (Amanda, Erlyn y Erick), a toda mi familia en general que es el gran TESORO que me queda.

Agradecer mi chica especial (Leidanis Hernández), la que conocí en mi último año de carrera y que me dio más del apoyo y amor del que necesitaba.

Agradecer también a mis amigos de infancia (Yoandry, Eiqui, María del Carmen, Maricela, Leosdany, Alberto, Osbel, David, Dari) y a los nuevos que conocí en la universidad (Hubert, Abel Cardona, los del piquete UCI en Masó, los de mi apto, a los de los grupos en los cuales estuve estos cinco años), ya que todos me apoyan cuando en realidad los necesita.

A mi compañero y amigo de Tesis (Ernesto Mató), por dedicar su tiempo y esfuerzo al nuestro Trabajo de Diploma.

A los del proyecto (Migue, Leidy, Yusmara, Virtudes, Olga, Pedro, Omar), a los profesores, que me impartieron clases, todos ellos jugaron un papel muy importante en el desarrollo y avance mis estudios en la universidad.

A mis vecinos (Angelina, Lily, Blanca, Sandra, Verdecia, Jacinto) que siempre se preocuparon y me dieron sus consejos para salir adelante y lograr mis objetivos en el transcurso de mis estudios por las diferentes enseñanzas

A los mundiales de las asignaturas que tuve y a el que me hizo aprobarlos, por mostrarme que para lograr algo en la vida hay que estudiar desde el inicio y no dejarlo

para el final.

En general agradecer a todos los que forman parte de mi vida, y a la que no se puede dejar de mencionar nunca a nuestra REVOLUCION, por hacer de sus hijos los mejores en el mundo. A todos ellos.

Ariel Ramírez Ferriol

A mis compañeros y amigos que han sido una verdadera familia en estos cinco años, junto a ellos he pasado momentos malos y buenos:

Yarenis, Leydis, Gloria, Pedro Luis, William, Alexis, Francisco (Collado), Jean Pablo, Máximo, Pedro Enrique, Luis, Gladys, Aimé y Fidel.

A mi compañero de tesis, que a pesar de todo lo complicado que estuvo siempre estuvo presente.

A los pipos como nos decimos cariñosamente, Juan Pablo, Fernando, Alexei y Yanet a ellos les agradezco toda la ayuda que me dieron, por aguantar mis malcriadeces y siempre estar ahí cuando los necesité. No se puede olvidar la comida de todos los días y a quien le tocaba fregar.

A mis tutores Pedro, Omar y Leidy, ellos supieron ser pacientes, comprensivos y darme respuesta a todas mis preguntas. Pedro gracias por tu tiempo por ser no solo tutor, también amigo y a Dailin por las salsas de perro.

Al grupo de trabajo de mantenimiento con ellos aprendí muchas de las cosas que posibilitaron hacer realidad este sueño.

Al grupo de trabajo de CRM, por las veces que me ayudaron cuando lo necesité.

Al tribunal que supieron hacer de mi un mejor profesional, siempre señalándome las cosas constructivamente, a la presidenta por insertarme en el mundo de la arquitectura de software y atender todas mis dudas cuando fue necesario.

A todos muchas gracias, han ayudado con su granito de arena que este sueño, que es el mío y el de mis padres hoy se hiciera realidad.

Ernesto Mató Roque

Dedicatoria

Dedico esta tesis a las personas meritorias de todos mis logros en esta vida, mi principal motivación, a los que me apoyan en estos años que llevo y llevaré de vida, a los que me dieron las fuerzas para superar los malos impedimentos, a los que me brindaron amor y educación: mis padres, mi familia, mis amigos, a los que tuve el placer de conocer una vez y me llevo gran recuerdo de ellos. En general a todos los que de una forma u otra ocupan un lugar importante para mí.

Ariel Ramírez Ferriol

En primer lugar a mis padres por darme tanto amor, confianza y apoyo en toda mi vida y principalmente en estos cinco años de carrera, sin su apoyo incondicional mi sueño que es el de ellos también no se hubiese realizado. Para ellos mi eterna gratitud y todo mi amor.

A mis abuelos, que han sido para mí los segundos padres, los que han sabido cuidarme cuando mis primeros no han estado por una razón u otra. A ellos todo el amor que se merecen.

A mis tías que siempre han sido muy atentas y preocupadas conmigo.

A mis primos que de una forma u otra han sido los hermanos que no tuve.

Ernesto Mató Roque

Resumen

En el presente trabajo se realiza un estudio detallado de diferentes conceptos relacionados con la arquitectura de software y las tecnologías, para el desarrollo de un Sistema de Administración de Relaciones con el Cliente (CRM, siglas en inglés), debido a que en estos instantes la solución tecnológica Cedrux, que surge con el objetivo de eliminar los problemas de gestión de recursos empresariales existentes en las entidades cubanas, no cuenta en su solución con la gestión de relaciones con el cliente. Esta solución está enfocada a mejorar las relaciones de las empresas con los clientes, para lograr una mejor productividad. El desarrollo del trabajo tiene como principal objetivo la definición de la línea base de un sistema CRM, que permita la integración con Cedrux para tener un mejor control de la información que se generan en las diferentes áreas de una empresa como: Mercadotecnia, Compra/Venta y Servicios.

La integración de los sistemas permite a las empresas cubanas que usan el ERP Cedrux, extender y explotar los beneficios de un CRM. Su extensión posibilita la valoración de las oportunidades de venta, la ampliación de la visión financiera del cliente, la mejora del trabajo en las áreas comerciales y de mercado de las empresas.

Para llevar a cabo esta solución se describen las vistas arquitectónicas a partir del modelo arquitectónico propuesto por el Centro de Informatización de la Gestión de Entidades (CEIGE), el cual brinda una explicación detallada de cada vista. El trabajo concluye sometiendo la solución a las pruebas del software mediante un método para el análisis de arquitectura de software, además de un análisis del resultado de estas pruebas.

Palabras claves: Arquitectura de Software, Interoperabilidad, Sistema de Administración de Relaciones con el Cliente.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

La gestión de la información juega un papel fundamental en el desarrollo del sector empresarial. Un manejo más eficiente de la misma se logra a partir del uso de las Tecnologías de la Información y las Comunicaciones (TICs). Con el desarrollo de estas han surgido modelos de gestión para las organizaciones, basados principalmente en la orientación al cliente, enfocándose en la optimización de los procesos. Por esta razón, surgen las estrategias de negocio enfocadas a seleccionar y gestionar los clientes. Con esto surgen los CRM capaces de administrar de forma eficiente un gran volumen de información, los cuales tienen como principal reto seguir atrayendo a clientes nuevos y rentables, al tiempo que se estrechan los lazos con los ya existentes para optimizar estas relaciones en todo su ciclo de vida (Vásquez, 2009).

Las empresas competitivas, en la actualidad, han detectado que el éxito hay que buscarlo en el manejo de una exitosa relación con el cliente. Un 12% de las empresas europeas utilizan una solución CRM, paralelamente, el 6% de las organizaciones europeas están en fase de implantación de soluciones CRM. El 72 % de las empresas consideran que la satisfacción de sus clientes es el reto más importante al que se enfrentan en los próximos años, de acuerdo con un estudio reciente de Dataquest (Vásquez, 2009).

La rentabilidad empresarial, viene con el cumplimiento de los siguientes objetivos: la interacción con el cliente, reconocer su valor actual, saber qué productos son de su preferencia, escuchar sus quejas y sugerencias y saber cómo usar toda esta información en beneficio de la empresa. Para el cumplimiento de estos objetivos, surgieron los CRM.

Actualmente, los CRM son uno de los temas que mayor atención ha generado en los campos de la estrategia de negocios, tecnología de la información y en la gestión de mercadotecnia. Consiste en comprender a los grupos de clientes y tratar a cada uno de forma tal que se maximice su valor, recurriendo habitualmente al uso de software que apoyen y faciliten el proceso (Activa, 2011). “El principal beneficio de un CRM para una empresa, es que permite a sus clientes recibir un mejor servicio a través de una atención personalizada, al tiempo que le ofrece la posibilidad de identificar nuevos clientes y mantener más satisfechos a los ya existentes (Vásquez, 2009)”. “Cuesta un

80 por ciento menos conservar un cliente actual que atraer a uno nuevo" (Vásquez, 2009).

Con una solución CRM las empresas pueden establecer un diálogo continuo con sus clientes utilizando la web para comunicarse con ellos directamente, conocer mejor sus necesidades y ofrecerles soluciones personalizadas, maximizar la eficacia de sus iniciativas gracias a la información que la empresa tiene de su cliente, conectar departamentos, permitiéndoles acceder a la misma información actualizada en tiempo real, dirigirse al cliente de un modo coherente desde cualquier punto de la estructura de la empresa (Vásquez, 2009).

El proyecto ERP-Cuba comenzó a desarrollarse en la Universidad de las Ciencias Informáticas en el 2008 con vistas a la obtención del producto Cedrux, como solución tecnológica a los problemas de gestión de recursos empresariales existentes en las entidades cubanas (Rivero Alvarez Tahirí, 2010). La obtención de este producto se materializó en el 2011 cuando el CEIGE, libera la primera versión de Cedrux, la cual consta con un conjunto de componentes que apoyan las principales actividades de una entidad.

Liberada esta versión, los profesionales del CEIGE quieren lograr el desarrollo de un sistema CRM, el cual se pueda integrar con el sistema Cedrux, dado que en estos instantes este no cuenta con una solución CRM y las empresas cubanas necesitan una solución que gestione las relaciones con los clientes para manejar: sus características, sus tendencias y su fidelidad hacia estas. En varios talleres realizados con uno de los equipos de desarrollo del departamento Soluciones Empresariales (SOLEM), perteneciente al CEIGE, el jefe de departamento y el director del centro CEIGE identificaron los siguientes problemas.

Ineficiencia a la hora de llevar a cabo algunas tareas, por ejemplo el personal de ventas consume demasiado tiempo en tareas administrativas, elaborando informes de visitas, informe de clientes y seguimientos. Por otro lado se tiene al personal de mercadotecnia, que tiene que realizar un mayor esfuerzo a la hora de llevar a cabo un estudio del mercado y lograr seleccionar el más factible para la empresa. En otra área se encuentran los trabajadores de servicios en función de la satisfacción de los clientes según los servicios prestados, a los cuales les resulta engorroso llevar un control y seguimiento de la documentación y plantillas de cada persona, para ejecutar

un servicio. Para darle respuesta a estos problemas ante la necesidad de que el centro cuente con un CRM se establece como cliente a la dirección del propio centro.

Estas dificultades son generadas por la falta de organización de la información que se encuentra en agendas personales e informes en formato duro, deficiencias que provocan un mayor tiempo de respuesta en la atención a los clientes y que influyen negativamente en la toma de decisiones que beneficien a la empresa. Todos estos inconvenientes propician la demora en la atención a solicitudes, incumplimientos de los contratos, pérdida de clientes debido a que no se gestiona la relación empresa-cliente y se toman decisiones erróneas por la falta de información. La solución arquitectónica propuesta debe soportar el desarrollo de un sistema CRM en el período de un año, permitiendo la interoperabilidad con Cedrux, la escalabilidad en el incremento de sus funcionalidades e incluir mecanismos de seguridad que prioricen el manejo de información sensible de las entidades.

Por lo antes expuesto se plantea como **problema a resolver:** ¿Qué definición y configuración de base tecnológica permitiría desarrollar un sistema para la gestión de relaciones con el cliente, siendo este escalable, seguro e interoperable con Cedrux?

Para la solución del problema planteado el **objeto de estudio** trazado lo constituyen los sistemas de gestión empresarial, quedando enmarcado en el **campo de acción** la definición arquitectónica de los sistemas de gestión de relaciones con el cliente.

Para la realización de este trabajo se plantea como **objetivo general:** Definir la línea base de un sistema para la gestión de relaciones con el cliente de manera que sea escalable, seguro e interoperable con el sistema Cedrux.

Para dar cumplimiento al objetivo general se trazan los siguientes **objetivos específicos:**

- Seleccionar las características más deseables de un sistema CRM, analizando la arquitectura de sistemas semejantes.
- Describir la línea base de la arquitectura.
- Evaluar arquitectónicamente la propuesta para determinar si la solución cumple con las necesidades del proyecto.

Idea a Defender

Definir la base tecnológica y su configuración para el desarrollo de un sistema de Administración de Relaciones con el Cliente contribuirá a que las empresas tengan un soporte informático que permita desarrollar los procesos de gestión de relaciones con el cliente, siendo este escalable, seguro e interoperable con Cedrux.

Se utilizarán como **métodos de investigación científica** los analizados a continuación:

Analítico – Sintético: Para analizar el sistema descomponiéndolo en partes, de forma que se pueda determinar cuáles son sus componentes esenciales y las relaciones entre ellos, con el objetivo de lograr un mejor entendimiento y poder definir las deficiencias y mejoras que conducen a la obtención del resultado esperado.

Histórico – Lógico: Para determinar cómo ha evolucionado el sistema, cuál es su lógica interna y la teoría en la que se basó su desarrollo.

Entrevista: Realización de entrevistas a los arquitectos del sistema Cedrux, para profundizar en el conocimiento y entendimiento sobre la arquitectura de este.

Observación: Se realizará un estudio de diferentes arquitecturas y de algunos sistemas, obteniendo una base y un mejor entendimiento.

Experimentación: Se realizarán pruebas de la arquitectura, mediante un método de evaluación de arquitecturas de software.

Modelación: Se realizará la creación de los artefactos necesarios para el desarrollo de la línea base de la arquitectura.

El documento está constituido por tres capítulos estructurados de la siguiente forma:

Capítulo1: Se expone una fundamentación teórica de la investigación, se realiza un estudio del estado del arte y se hace referencia a los conceptos fundamentales de la investigación para permitir entender cualquier concepto asociado a la comprensión del trabajo.

Capítulo2: Se expone la solución a la situación problemática previamente descrita haciendo uso del modelo arquitectónico propuesto por el CEIGE.

Capítulo3: Se somete la solución a pruebas de software a través de un método de evaluación de arquitecturas de software, que incluye las pruebas de interoperabilidad, seguridad y escalabilidad concluyendo con el resultado de las pruebas.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se incluyen términos y conceptos relacionados con CRM, arquitectura de software y elementos fundamentales que la definen como: los estilos y vistas arquitectónicas así como atributos de calidad. Se realiza además un análisis de diferentes CRM y de sistemas de planeación de recursos empresariales (ERP, siglas en inglés) que cuentan con un CRM, teniendo en cuenta su tecnología y su arquitectura. Se efectúa un estudio de algunos servidores de mensajería instantánea y herramientas de inteligencia de negocio. Todo esto se lleva a cabo para seleccionar las características necesarias a tener en cuenta para el desarrollo de un sistema CRM.

1.2 Sistema de administración y relaciones con el cliente CRM

Rigby et al. 2002 proponen que CRM consiste, simplemente, en comprender a los grupos de clientes y tratar a cada uno de una forma que se maximice su valor, recurriendo habitualmente al uso de software que apoyen y faciliten el proceso (Vásquez, 2009).

Para Croteau y Li 2003, CRM es un concepto que permite a una organización confeccionar productos y servicios específicos para cada cliente individual (Vásquez, 2009).

Por su parte, Piccoli et al. (2003) entienden CRM como una filosofía de gestión que permite a la empresa llegar a establecer relaciones más familiares con sus clientes (Vásquez, 2009).

Según lo expresado anteriormente se puede decir que los CRM son una estrategia de negocio centrada completamente en los clientes, mejorando cada día la relación empresa-cliente. El presente trabajo se regirá por el concepto expresado por Rigby, dado que este aborda desde las buenas relaciones con los clientes, hasta la creación de un software que lo permita. Se hace necesario el estudio de algunos aspectos relacionados con la arquitectura de software, debido que esta es la encargada de darle la estructura al software (Camacho, et al., 2004).

1.3 Arquitectura de software

La Arquitectura de Software (AS) es un concepto bastante joven, tiene sus orígenes en la década de los 60 cuando fue identificado en el trabajo de investigación de Edsger'-Dijkstra en 1968 y David Parnas a principios de los años 70. Este campo aumentó su popularidad desde los años 90 que Dewayne Perry y Alexander Wolf la exponen en el sentido que hoy se conoce.

En la actualidad se puede encontrar gran diversidad de conceptos relacionados con la AS, debido a que no todos los arquitectos tienen la misma concepción de este tema por lo que cada cual asume su propia interpretación.

Una definición reconocida es la de Clements: "La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones" (Reynoso, 2004). Se evidencia la importancia que tiene la definición de los componentes bases para el éxito de un sistema y la integración entre ellos, también se observa la utilidad de las vistas de la arquitectura, dado que estas facilitan un mejor entendimiento de la aplicación.

En una definición tal vez demasiado amplia, David Garlan establece que la AS constituye un puente entre el requerimiento software y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño (Reynoso, 2004). En este concepto se manifiesta el gran peso que tiene definir una arquitectura para la construcción de un sistema, debido a que esta transforma los requisitos en la estructura de la aplicación.

Otra de las definiciones de AS es la que brinda el documento de Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, siglas en inglés) Std 1471-2000, adoptada también por Microsoft, el cual define que: "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución" (Reynoso, 2004). En esta definición se demuestra una vez más el papel que juega para el desarrollo de un sistema.

Según estos conceptos expuestos se puede afirmar que la arquitectura de software es la base fundamental que guía o conduce el desarrollo de un sistema mostrando la estructura y el comportamiento del mismo y la relación que existen entre sus componentes. Después del análisis de las definiciones anteriores se decide que el concepto a utilizar es el expresado por Clements, dado que expresa una definición bastante amplia de AS, mientras que los demás autores expresan la definición a grandes rasgos.

Estilos arquitectónicos

Mary Shaw y Paul Clements identifican los estilos arquitectónicos como un conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer en sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo (Reynoso Carlos, 2004).

Roy Fielding sintetiza la definición de estilo, diciendo que un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles/rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo (Reynoso Carlos, 2004).

En el artículo “Estilos y Patrones en la Estrategia de Arquitectura de Microsoft” existe diversidad de grupos de estilos dada por diferentes autores entre estos se encuentran (Reynoso Carlos, 2004): Estilos de Flujos de Datos, Estilos Centrados en Datos, Estilos de Llamada y Retorno, Estilos de Código Móvil, Estilos Heterogéneos, Estilos Peer-to-Peer. Cada uno de estos estilos reúne un conjunto de estilos arquitectónicos, algunos de estos serán descritos más adelante.

A continuación se expone una breve descripción acerca de la familia de estilos arquitectónicos llamada y retorno, esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala (Reynoso Carlos, 2004). Miembros de la familia son:

Arquitectura en capas: En este estilo arquitectónico cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás (Reynoso Carlos, 2004).

Arquitectura orientada a objetos: Los componentes de este estilo son los objetos, o más bien instancias de los tipos de datos abstractos. En la caracterización clásica de David Garlan y Mary Shaw 1994, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos (Reynoso Carlos, 2004).

Arquitecturas basadas en componentes: Hay un buen número de definiciones de componentes, pero Clemens Alden Szyperski proporciona una que es bastante operativa: un componente de software, dice, es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas (Reynoso Carlos, 2004).

En este estilo el sistema será descompuesto en componentes, promoviendo el desarrollo y utilización de componentes reutilizables.

Modelo vista controlador: El estilo modelo vista controlador separa los datos de una aplicación, la vista del usuario y la lógica de control en tres clases diferentes. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). La vista maneja la visualización de la información, y el controlador interpreta las acciones, informando al modelo y/o a la vista para que cambien según resulte apropiado (Reynoso Carlos, 2004).

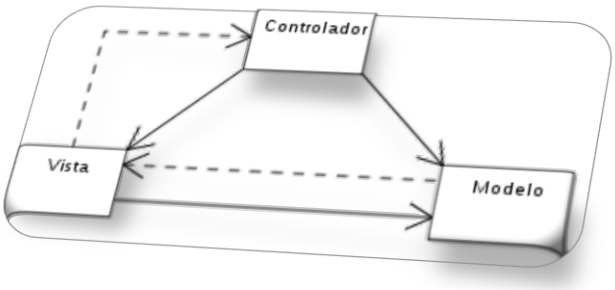


Figura 1 Estructura del estilo MVC.

Se puede decir que los estilos son reglas, las cuales completan los elementos fundamentales de AS, estos son componentes, conectores, configuraciones y

restricciones. Estos facilitan la reutilización, cuando se cuenta con un sistema similar al que se pretende desarrollar y que haya usado alguno de estos estilos. Sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones. Mediante ellos se puede lograr la evaluación de arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes requisitos. Esto facilitará comprender con mayor claridad algunos conceptos arquitectónicos que se tratarán a la hora de llevar a cabo el análisis de algunos sistemas CRM.

Vistas de la arquitectura de software

Buschmann establece que una vista arquitectónica representa un aspecto parcial de una arquitectura de software, que muestra propiedades específicas del sistema haciendo uso indistinto de los términos estructura y vista, proponen que las estructuras arquitectónicas pueden definirse agrupando los componentes y conectores de acuerdo a la funcionalidad del sistema, sincronización y comunicación de procesos, distribución física, propiedades estáticas, propiedades dinámicas y propiedades de ejecución, entre otras (Camacho, et al., 2004).

Por su parte, Kruchten define una vista arquitectónica como una descripción simplificada o abstracción de un sistema desde una perspectiva específica, que cubre intereses particulares y omite entidades no relevantes a esta perspectiva (Camacho, et al., 2004).

En la universidad la Infraestructura Productiva (IP) definió el conjunto de vistas arquitectónicas que deben desarrollarse para construir sistemas de software estas son: vista tecnológica, incluye (vista de tecnología, vista de seguridad y vista de presentación), vista de sistema, incluye (vista de sistema, vista de integración y vista de datos) vista de infraestructura, incluye (vista de desarrollo y vista de despliegue) además se le agregará una vista de procesos. Fueron definidas por un conjunto de especialistas que asumieron el “Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestión” (Ochoa, 2011) que las contempla. El CEIGE propone un refinamiento de estas vistas, constituyendo un subconjunto de lo propuesto en el modelo mencionado. Por lo tanto se tomará el modelo arquitectónico propuesto por el CEIGE, que propone una descripción más sencilla de las vistas arquitectónicas y se encuentra vigente en el CEIGE.

El refinamiento del modelo incluye los siguientes artefactos: complejidad y criticidad, especificación de componentes, línea base de la arquitectura, matriz de integración entre componentes y matriz de integración entre sistemas.

Atributos de calidad

Según Kazman los atributos de calidad son requerimientos adicionales del sistema que hacen referencias a características que este debe satisfacer, diferentes a los requerimientos funcionales (Camacho, y otros, 2004).

A grandes rasgos, se establece una clasificación de los atributos de calidad en dos categorías:

Observables vía ejecución: aquellos atributos que determinan el comportamiento del sistema en tiempo de ejecución (Camacho, y otros, 2004).

No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema (Camacho, y otros, 2004).

1.4 Sistemas CRM

Microsoft Dynamics es un sistema flexible proporciona a los usuarios la capacidad de crear una vista global de cada cliente, desde el contacto inicial hasta la post-venta y el servicio. Es 100% .Net, para el desarrollo de este sistema se utilizó C#, como gestor de base de datos se utilizó SQL Server con arquitectura orientada a servicios (SOA, siglas en inglés) y multicapas consistente en un cliente basado en web (Microsoft, 2008).

Entre las ventajas que brinda se encuentra la posibilidad de ser integrable con otras aplicaciones de oficina, está disponible en 22 idiomas incluyendo español e inglés. Posee módulos como: mercadotecnia, ventas y servicios. Permite un sencillo acceso a la información de sus clientes, dar alta y organizar nuevos contactos y hacer un seguimiento de las oportunidades actuales.

Entre las desventajas que posee es la no visibilidad del código de esta herramienta al ser propietaria y por lo tanto no se encuentra entre las políticas de software libre que persigue la universidad en el desarrollo de sus sistemas. Necesita tecnología propietaria para funcionar (C#, SQL Server). Solamente se puede ejecutar en sistema

el operativo de Windows, o sea, no es multiplataforma por lo que impone un pago altamente caro (Microsoft, 2008).

SugarCRM es un CRM construido con tecnología de código abierto, sin licencia para el usuario y desarrollado con tecnología web. Se ejecuta sobre apache, como lenguaje de programación utiliza PHP, sobre base de datos MySQL u Oracle. Utiliza tecnologías aunque modernas, confiables como: XML, notación de objetos JavaScript (JSON, siglas en inglés), AJAX, SOAP, con arquitectura cliente/servidor (Fredricks, 2009).

Entre las ventajas que brinda es que es multiplataforma, se puede acceder desde múltiples versiones de navegadores algunas de ellos son: (Internet Explorer, Mozilla Firefox, Apple Safari). Es adaptable a la infraestructura existente en cualquier empresa. Tiene una interfaz amigable, sin necesidad de una búsqueda compleja y está disponible en diferentes idiomas entre los que se encuentran el inglés y el español. Facilita la integración con aplicaciones de oficina y otras como Thunderbird, Zucker Reports, Asterisk (Fredricks, 2009).

Esta herramienta maneja un gran número de módulos funcionales, sin embargo los principales para las empresas son de carácter comercial, lo que implica que para obtenerlo hay que pagar, lo mismo sucede con los plugins y actualizaciones. Actualmente presenta algunas deficiencias como: crear nuevas pestañas y nuevos módulos esto traería problemas para futuras actualizaciones.

ZOHOCRM es un sistema de administración de relaciones con el cliente, está basado en una arquitectura completamente web. Este se ejecuta sobre apache y como lenguaje de programación se utilizó PHP, gestor de base de datos MySQL y utiliza el servidor de protocolo para comunicación en tiempo real (XMPP, siglas en inglés) (Andina, 2009).

Entre las ventajas que brinda es que es multiplataforma, se puede acceder desde diferentes navegadores web algunos de ellos son: (Firefox, Opera, Internet Explorer), disponible en diferentes idiomas entre los que se encuentra el inglés y el español. Entre sus desventajas, se encuentra que es sobre demanda, se paga solo para los usuarios que lo necesiten, es gratuito hasta un máximo de tres usuarios y es de código cerrado, no cumpliendo con las políticas de la universidad (Andina, 2009).

VTigerCRM es un sistema para pequeñas y medianas empresas, está construido con

una tecnología rápida y confiable. Es de código abierto o sea no hay pago de licencias de uso, y permite la adecuación de la herramienta a necesidades específicas de la empresa, se encuentra en varios idiomas, es modular y con un ambiente web. Se ejecuta sobre apache, y como lenguaje de programación utiliza PHP, Javascript y Visual Basic. Tiene soporte de base de datos múltiples (MySQL, PostgreSQL) (Postbank, 2012).

Entre las ventajas que brinda se encuentran que es multiplataforma. Este ofrece integración con Microsoft Office, Thunderbird y Outlook. Permite el intercambio de la información en tiempo real. Entre sus desventajas está, que tiene una interfaz muy cargada. No se le brinda soporte en Cuba y no cuenta con una comunidad que trabaje en mejorar el sistema según las necesidades de los usuarios (Postbank, 2012).

SalesForce es un CRM, que permite gestionar la comunicación de la empresa y sus clientes, y ofrece integración con los sistemas de gestión de tareas y email más habituales, como Outlook, GMail. Este está desarrollado sobre la plataforma Force de SalesForce. Es completamente basado en web, provee servicios para operaciones de ventas, marketing y centros de llamadas que simplifica la gestión de las relaciones con los clientes. Como lenguaje de programación PHP, Ruby onRails, Perl, lenguaje de marcado para hipertextos (HTML, siglas en inglés), Javascript, C/C++, Win32, .Net, J2EE, ASP y como gestores de base de datos puede usar cualquiera de los siguientes: (MySQL, PostgreSQL, Oracle, Microsoft SQL). Con una infraestructura multiusuario siguiendo los principios de SOA (Salesforce, 2008).

Este es un excelente CRM, pero el mismo está desarrollado sobre una plataforma propietaria lo que implica que su uso es únicamente de carácter comercial, no facilitando la independencia tecnológica, otro de los problemas que presenta, es que no es de código abierto (Salesforce, 2008).

Se han analizado algunas de las características de los diferentes sistemas CRM encontrando similitudes entre ellos. Se puede apreciar que todos poseen un entorno web, utilizan los estilos arquitectónico llamada y retorno y otros Peer to Peer, siendo estos unos de los estilos más usados en la actualidad debido a las posibilidades que brindan de poder desarrollar aplicaciones con filosofía cliente/servidor. También permiten la integración con diferentes herramientas de ofimática y poseen varios idiomas como el español e inglés.

Poseen algunos inconvenientes como: Microsoft Dynamics y SalesForce son propietarios, implicando gastos en licencia, ZOHOCRM permite un máximo de tres usuarios, después de esta cantidad será pagada su licencia, SugarCRM tiene la inconveniencia que el gestor de base de datos que utiliza es propietario, VTigerCRM presenta como inconveniente no brindar soporte para Cuba.

1.3 Sistemas ERP con CRM

OpenERP es un sistema de gestión empresarial, de licencia publica general (GPL, siglas en inglés) que cubre las necesidades en el área de CRM, dado que este cuenta con diferentes componentes relacionados, como mercadotecnia, compra/venta y servicios. Posee una arquitectura cliente/servidor, permitiendo la comunicación entre el cliente y el servidor mediante el estándar NET-RPC y XML-RPC, este facilita al cliente la llamada de procedimientos remotos. Es multiplataforma, utiliza como gestor de base de datos PostgreSQL, está desarrollado íntegramente en python y proporciona una interfaz de usuario completamente web, utilizando para esta XML. Utiliza la familia de estilos llamada y retorno. (Catalá Gil Sergio, 2009).

Uno de los problemas que este presenta es que no cuenta con un módulo, ni con una herramienta en concreto que maneje BI. Cuenta con un flujo de trabajo flexible y dinámico, una de las grandes ventajas con las que cuenta OpenERP es el uso de OpenObject este es un marco de trabajo, que cuenta con un proceso denominado desarrollo rápido de aplicaciones, proporcionando un desarrollo acelerado, además utiliza el estilo arquitectónico MVC (Catalá Gil Sergio, 2009).

Openbravo es un sistema de gestión empresarial, totalmente basado en web, está creado a partir de código abierto, desarrollado en java siguiendo la filosofía de MVC. Se ejecuta sobre apache y tomcat, soporta base de datos Oracle y Postgres. Este utiliza tecnología confiable como JavaScript, SQL y PL/SQL, XML, HTML con arquitectura cliente/servidor y utiliza la familia de estilos llamada y retorno. Es un sistema adaptable, modular, y de base de datos centralizada, posee funcionalidades en el área de CRM, dado que tiene una zona de ventas en la que gestiona, pedidos de ventas, facturas, etc. Consta con un módulo que maneja BI, donde se definen los cuadros de mando e indicadores claves sobre la actividad de la empresa. Permite la integración con sistemas de gestión de contenido, herramientas de BI etc. Su usabilidad es pobre y aunque el software es gratuito su implantación es costosa (Catalá Gil Sergio, 2009).

OpenXpertya es una solución empresarial, basada en software libre que incluye solución CRM, dado que cuenta con los procesos de compra/venta, cuenta también con mensajería interna posibilitando el aviso a los clientes. Este es un sistema multiplataforma, puede utilizar cualquiera de los siguientes gestores de base de datos (Oracle, PostgreSQL, Firebird, Sybase, etc), como lenguaje de programación utiliza java (Eclipse IDE) y como familia de estilos arquitectónicos llamada y retorno. Consta con una solución de procesamiento analítico en línea (OLAP, siglas en inglés) que permite la explotación de las base de datos. Se basa en una arquitectura cliente/servidor mediante un modelo original en tres capas, en la capa de datos se encuentra el motor de base de datos relacional, en la capa de aplicación se encuentra el servidor de aplicaciones JBoss basado en java y las clases que interactúan directamente con la base de datos vía conectividad de base de datos java (JDBC, siglas en inglés) y en la capa de presentación dispone de varios clientes posibles. Una de las deficiencias que tiene es el uso de JNPL dado que esta es propietaria de la empresa Sun Microsystems. (Catalá Gil Sergio, 2009).

CompireERP está desarrollado completamente sobre java. Utiliza como base de datos Postgres u Oracle y como servidor web JBoss. Se ejecuta bajo Licencia Pública Compire (CPL, siglas en inglés), pero realmente es difícil saber cuánto del producto es de código abierto y cuánto no, al incluir varias librerías internas cuyo código no se proporciona con el producto e incluso algunas de pago que realizan funciones centrales en el sistema. Se basa en una arquitectura cliente/servidor utilizando la familia de estilos llamada y retorno, se comunica mediante la conectividad de base de datos en java (JDBC, siglas en inglés) con la base de datos mediante la invocación de métodos remotos (RIM, siglas en inglés) con el servidor de aplicaciones, se puede ejecutar sobre Unix, Linux, MacOS X y Windows (Catalá Gil Sergio, 2009).

Este posee funcionalidades en el área de gestión comercial que proporcionan determinación a la hora de intercambiar con un cliente. Además no consta con un módulo independiente sino una vista lógica de todas las actividades relacionadas con el cliente o sea es una parte integral del proceso de negocio, consta también con una solución OLAP para el análisis de grandes cantidades de datos en una base de datos. Este necesita tecnología propietaria para su funcionamiento librería de ficheros PDF y bibliotecas de Sun Microsystems (Catalá Gil Sergio, 2009).

SAP es un sistema empresarial en utiliza como lenguaje de desarrollo programación de aplicaciones de negocios avanzados (ABAP, siglas en inglés), este lenguaje de programación fue creado por la misma empresa SAP. Utiliza sentencias Open SQL, posibilitando poder conectarse prácticamente a cualquier base de datos, en la capa de presentación utiliza web Dynpros, para la conexión con ABAP, J2EE y .NET. Se basa en una arquitectura cliente/servidor de tres capas, la capa de base de datos, la capa de aplicación y la capa de presentación. De esta manera el cliente solicita un servicio ofrecido por el servidora través del canal de comunicación: LAN o WAN (Hernández, 2008).

Este tipo de arquitectura puede tener múltiples configuraciones, la primera es un sistema centralizado, donde los tres niveles mencionados se encuentran en la misma computadora, la segunda es un sistema de dos niveles, en este esquema un servidor ejecuta el sistema de gestión de base de datos (DBMS, siglas en inglés), nivel de almacenamientos de datos y el nivel de aplicaciones, mientras otro servidor es el encargado del nivel de presentación. Este último servidor es el encargado de controlar las entradas y salidas del sistema R/3. Por último está la configuración en tres niveles, esta se logra cuando cada uno de los niveles mencionados se encuentran en su propia plataforma de ejecución, también utiliza el MVC (Hernández, 2008).

SAP permite la integración mediante una tecnología creada por ellos mismos conocida como llamada de funciones remotas (RFC, siglas en inglés), también utiliza el enlace de aplicaciones habilitadas (ALE, siglas en inglés), acoplando el intercambio de información entre otros sistemas, asegurando la consistencia y seguridad de los datos. SAP posee una solución CRM para medianas y pequeñas empresas, cuenta con componentes como mercadotecnia, ventas y servicios (Hernández, 2008).

Tabla 1 Comparativa de ERP con CRM.

Características/Sistemas	SAP	OpenERP	Openbravo	OpenXpertya	CompireERP
Licencia	SAP	GPL	OBPL	LPO	CPL
Entorno	Web	Web	Web	Web	Web
Estilo arquitectónico	Peer to Peer, Llamada y retorno	Llamada y retorno	Llamada y retorno	Llamada y retorno	Llamada y retorno
Filosofía	Cliente/Servidor	Cliente/Servidor	Cliente/Servidor	Cliente/Servidor	Cliente/Servidor
Multiplataforma	SI	SI	SI	SI	SI
Base de Datos	MSSQL, Oracle, DB2, MySQL	PostgreSQL	Oracle, PostgreSQL	Oracle, PostgreSQL, Firebird, Sybase	Oracle PostgreSQL
Lenguaje de programación	ABAP	Python	Java	Java	Java
Workflow	SI	SI	SI	SI	SI

Los datos mostrados en la tabla comparativa sobre los distintos sistemas estudiados cuentan con similitudes en algunos indicadores, por lo que se decide que el sistema a desarrollar posea las mismas similitudes siendo estos: el uso de la familia de estilos llamada y retorno con filosofía cliente/servidor, multiplataforma y el uso del gestor de base de datos PostgreSQL. Otra de las similitudes es la contención de un motor de flujo de trabajo y lenguaje de programación java, sin embargo, estos sistemas que utilizan el lenguaje de programación java como:

Openbravo, solo publica parte del código de la versión Network de Openbravo, además la licencia no cubre el sistema en su totalidad, sino que hay partes con licencias privativas diversas y tampoco permite enviar facturas por correo electrónico.

CompireERP para su funcionamiento necesita tecnología propietaria. **OpenXpertya** para actualizarse a nuevas versiones, es necesario descargar el producto completo y volverlo a instalar, en lugar de poder gestionar dichas actualizaciones desde el mismo producto, actualizando solo aquellas partes del programa que hayan cambiado y en la capa de presentación utiliza tecnología propietaria. **SAP** es un excelente sistema, que cuenta con facilidades de integración y desarrollo, pero es propietario y para su uso impone un alto impuesto.

Lo anteriormente analizado permite decidir utilizar OpenERP, único sistema que utiliza licencia GPL cumpliendo con la política de desarrollo e independencia tecnológica abogada por Cuba. Es bastante flexible y simple mostrando gran facilidad para realizar modificaciones y adaptaciones del código según las necesidades y finalidades

deseadas por el usuario, cuenta con los componentes bien definidos para el desarrollo de un sistema CRM. Reutilizando los componentes del CRM que contiene y permitiendo la integración con el sistema Cedrux desarrollado en el CEIGE, facilitando los mecanismos de integración e intercambio de servicios con Cedrux a petición del cliente.

1.4 Mecanismos de integración entre sistemas

Protocolo para el acceso simple a objetos

El protocolo para el acceso simple de objetos (SOAP, siglas en inglés) es un recurso novedoso para el desarrollo de aplicaciones.

SOAP es altamente flexible y puede soportar diversas aplicaciones; sin embargo su mayor expresión se alcanza cuando se utiliza en la llamada a procesos remotos (RPC siglas en inglés) sobre protocolo de transferencia de hipertexto (HTTP, siglas en inglés) y XML. SOAP es un protocolo de invocación de objetos basado en lenguaje de marcas extensibles (XML, siglas en inglés) y ha sido aplicada en la comunicación de aplicaciones sobre HTTP y le permite funcionar detrás de los cortafuegos empresariales lo que potencia el acceso a servicios independientes de plataforma (Kosftikian, 2008).

Lenguaje de descripción de servicios web

El lenguaje de descripción de servicios web (WSDL, siglas en inglés), surgen en septiembre de 2000 de la mano de Microsoft, IBM y Ariba, constituye un estándar para la descripción de servicios del World Wide Web Consortium (W3C, siglas en inglés), organización que guía el desarrollo en la web (Endrei, 2012).

En esencia, WSDL es un contrato entre el proveedor del servicio y el cliente mediante el que el proveedor del servicio indica (Endrei, 2012).

- ✓ ¿Qué funciones que se pueden invocar?
- ✓ ¿Qué tipos de datos utilizan esas funciones?
- ✓ ¿Cómo acceder a los servicios? En esencia, mediante qué URL se utilizan los servicios.

1.5 Tecnologías

En el presente epígrafe se realizará un estudio de las diferentes tecnologías que posibilitan definir la base tecnológica, se analizarán librerías y servidores para la interacción con la mensajería instantánea brindando información en tiempo real, así como el estudio de herramientas de inteligencia de negocio para facilitar el manejo de la información.

1.5.1 Librerías para la interacción con mensajería instantánea

XMPPPY es una librería de python que está dirigida a proporcionar las secuencias de comandos fáciles con jabber. Esta librería no fue diseñada desde cero, hereda código de jabberpy y tienen una interfaz de programación de aplicaciones (API, siglas en inglés) muy similar, contiene un alto nivel de funciones para el envío de mensajes y puede crear sus mensajes en formato XML. Esta es de código abierto y utiliza el protocolo extensible de mensajería y comunicación de presencia (XMPP, siglas en inglés) (Sourceforge, 2012).

Esta librería analizada es compatible con OpenERP, ambas soluciones se implementan utilizando el lenguaje python, razón por la cual se recomienda como librería para la interacción con la mensajería instantánea de la solución.

1.5.2 Servidores de mensajería instantánea

Openfire es una poderosa plataforma de mensajería instantánea y servidor chat que implementa el protocolo XMPP/Jabber, escrito en java y provee licencias comerciales y licencia pública general (GNU, siglas en inglés). Openfire se destaca por su simplicidad a la hora de instalarlo y administrarlo, también destaca su flexibilidad a la hora de su personalización e integración con otras aplicaciones (Londoño Guzmán Jeissy Alexandra, 2009).

Las características principales son las siguientes (Londoño Guzmán Jeissy Alexandra, 2009):

- ✓ Servidor jabber implementado en java.
- ✓ Administración basada en una interfaz web amigable.
- ✓ Características adicionales desarrolladas e integradas a través de plugins.
- ✓ Plataforma independiente desarrollada puramente en java.

- ✓ Personalizable.
- ✓ Seguro, capa de conexión segura (SSL, siglas en inglés)/TLS.
- ✓ Almacenamiento en Active Directory, a través del protocolo de acceso a directorios livianos (LDAP, siglas en inglés), a gestores de base de datos tales como: MySQL, Oracle y PostgreSQL.
- ✓ Abierto e ínter operable con otras plataformas XMPP.
- ✓ Permite el registro de usuarios desde cliente jabber.

Tigase es un servidor XMPP/Jabber ligero y escalable, escrito en java. Se puede utilizar como aplicación integrada dentro de otro sistema. El bajo consumo de recursos hace que sea una buena solución para las pequeñas instalaciones y la escalabilidad hace que sea también bueno para los despliegues de muy alta carga y un enorme número de usuarios, puede instalarse en tantas máquinas como sea necesario. Es modular y extensible (Villacampa, 2008).

Las características principales son las siguientes (Villacampa, 2008):

- ✓ Código abierto y libre.
- ✓ Robusto y fiable.
- ✓ Seguro (SSL/TLS).
- ✓ Flexible.
- ✓ Extensible.
- ✓ Fácil de instalar y mantener.

OpenIM es un servidor jabber implementado en java y de código abierto. El propósito principal de OpenIM es proporcionar un servidor de mensajería instantánea, simple y altamente eficiente con una alta modularidad del código fuente (Villacampa, 2008).

Las características principales son las siguientes (Villacampa, 2008):

- ✓ Gran estabilidad.
- ✓ La integración con LDAP.
- ✓ La mayoría de las funcionalidades de mensajería instantánea están soportadas.
- ✓ Comunicación servidor a servidor.
- ✓ Seguro (SSL).
- ✓ Almacenamiento y registro de conversación, para estadísticas o supervisión.
- ✓ No permite salas de chat.

Jabberd14 es un servidor jabber implementado en C/C++. Jabber14 es la implementación original del protocolo jabber (Villacampa, 2008).
Las características principales son las siguientes (Villacampa, 2008):

- ✓ Servidor jabber implementado en C/C++.
- ✓ Soporte excelente para protocolos de seguridad y encriptación.
- ✓ Soporta otros protocolos, no solo el protocolo XMPP/Jabber.
- ✓ Cumple estrictamente las normas del protocolo jabber.
- ✓ Personalizable e integrable en sitios web.
- ✓ Gran comunidad de desarrolladores.

A continuación se presentará una tabla comparativa con las características más relevantes.

Tabla 2 Comparativa de servidores de mensajería instantánea.

	OpenFire	Tigase	Open IM	Jabberd14
Código Abierto	✓	✓		✓
Autenticación	✓	✓	✓	✓
Seguridad	✓	✓	✓	✓
Extensible	✓	✓	✓	✓
Administración Web	✓			
Escalable	✓	✓		✓
Robustez	✓	✓	✓	✓

Openfire será el servidor de mensajería instantánea elegido, es el único que soporta la administración web. Además es compatible con Windows, Linux, Mac OS, también permite la transferencia de archivo entre los usuarios, la compresión de datos, contiene una interfaz para agregar plugins y consta con un panel de administración.

1.5.3 Inteligencia de negocio

En este epígrafe se expondrán algunos conceptos relacionados con Business Intelligence (BI), Howard Dresner, fue el primero que acuñó el término en 1989. Éste, definió BI como:

"BI es un proceso interactivo para explorar y analizar información estructurada sobre un área, para descubrir tendencias o patrones, a partir de los cuales derivar ideas y extraer conclusiones. El proceso de BI incluye la comunicación de los descubrimientos y efectuar los cambios. Las áreas incluyen clientes, proveedores, productos, servicios y competidores" (Núñez, 2010). Se puede expresar que BI es un conjunto de

herramientas diseñadas para convertir datos en información y a su vez en conocimiento, mediante el análisis de un conjunto de datos que se disponen en una organización, para conducir de manera eficaz los procesos de negocio de las organizaciones.

El objetivo primario de BI es ayudar a las empresas a tomar decisiones que mejoren el rendimiento de la compañía e impulsen su ventaja competitiva en el mercado. Tomar mejores decisiones significa mejorar alguna o todas las partes del proceso, tomar un menor número de decisiones erróneas y un mayor número de decisiones acertadas (Rosa, 2010).

Para la definición arquitectónica del sistema a desarrollar, es importante su uso debido a que posibilitan la toma de decisiones mucho más acertadas, mejorando así la relación empresa-cliente. A continuación se describen algunos sistemas que posibilitan el manejo de BI. Del mismo modo BI pretende ser la vía para que las organizaciones y las sociedades obtengan nuevos conocimientos y hagan un mejor uso de la información que poseen.

Pentaho es un proyecto iniciado por una comunidad de código abierto, provee una alternativa de soluciones de inteligencia de negocio en distintas áreas como en la Arquitectura, Soporte, Funcionalidad e Implantación. Estas soluciones al igual que su ambiente de implantación están basados en java, haciéndolo flexible en cubrir amplias necesidades empresariales. A través de la integración funcional de diversos proyectos de código abierto permite ofrecer soluciones en áreas como: análisis de información, reportes, tableros de comando, flujos de trabajo y minería de datos. Este integra componentes de código abierto mostrando una combinación de flujos de trabajo y administración de procesos. Es una herramienta de reportes flexibles y con clase empresarial, de escritorios o basados en web. La herramienta de reportes Pentaho permite comenzar desde sencillos reportes iníciales hasta formar complejos reportes ajustados a las necesidades del negocio. Provee un completo conjunto de algoritmos que automatizan los procesos de transformación de datos a la forma en que la minería de datos puede explotarlos. Los resultados pueden ser visualizados en modo gráfico ya sean agrupados, segmentados, de árbol de decisión, bosques aleatorios, redes neurales y componentes de análisis (Stratebi, 2010).

JasterReports es desarrollado por la empresa JasperSoft, la herramienta creada por esta empresa, es de código abierto, proporcionando una solución completamente web,

utilizando java como lenguaje de programación. Al igual que Pentaho esta solución fue la integración de diferentes proyectos ofreciendo soluciones en diferentes áreas, análisis de información, reportes, tableros de comando y flujos de trabajos. Los tableros de comandos que contiene esta solución, no fueron desarrollados por la empresa fueron tomados de Pentaho. Posibilita exportar todas las aplicaciones que contiene como servicios web, propiciando la integración con otras aplicaciones. Los informes desarrollados con esta solución permiten ser exportados en formato PDF, HTML, XML, CSV, RTF, XLS y TXT. Su mejor solución son los informes ad-hoc, permitiendo la selección de diferentes tipos de plantillas y formatos, la selección de diferentes orígenes de datos y la creación de informes arrastrando los campos al lugar deseado (Stratebi, 2010).

Palo es el motor MOLAP implementado por la empresa Jedox. Esta herramienta es completamente web y de código abierto. En sus inicios comenzó a utilizar C++ como lenguaje de programación, después fue cambiando a java pero sin migrar el código que estaba desarrollado en C++. Este motor brinda soluciones en diferentes áreas como, reportes, tableros de comandos e informes. Los informes desarrollados en Palo, son realizados en hojas de cálculo que se cargan con los datos del motor MOLAP. Esta solución cuenta con un plugins, llamado PaloforExcel que es quién permite conectarse al motor de Palo y explorar los datos desde una hoja Excel (Stratebi, 2010).

BIRT esta herramienta es desarrollada por la empresa Actuate, proporcionando una solución completamente web. En sus inicios simplemente era un plugins para Eclipse, igual a otros surgió con la unión de diferentes aplicaciones. Propicia soluciones en áreas como: informes, exportación de diferentes tipos de documentos, integración con Eclipse y tableros de comando. Actualmente la herramienta permite realizar informes desde el servidor. El motor OLAP que tiene solo existe como motor interno para la realización de tablas cruzadas insertadas dentro de un informe y este motor no posee cuadros de mando (Stratebi, 2010).

Tabla 3 Comparativa de herramientas de BI (Stratebi, 2010)**.**

Características/Sistemas	Pentaho	JasterReports	Palo	BIRT
Extraer, Transformar, Cargar(ETL)	B	B	B	M
Integración de datos	B	B	B	M
Servidor de B.I	B	B	B	B
Informes	B	B	B	R
Informes predefinidos	B	B	B	R
Informes ad-hoc	R	B	M	B
Motor OLAP	B	R	B	B
Visor OLAP	R	R	B	M
Cuadros de mando	B	R	R	R
Cuadros de mando ad-hoc	M	M	M	M

De las herramientas estudiadas JasterReports, Palo y BIRT, presentan algunas deficiencias en áreas como: ETL, integración de datos, informes ad-hoc, visor OLAP y cuadros de mando ad-hoc. Se decidió el uso de Pentaho por ser el más completo, en su facilidad de uso, mantenimiento y flexibilidad a la hora de realizar transformaciones. Se integra con la mayoría de los entornos (Stratebi, 2010).

1.6 Conclusiones parciales

La investigación realizada arrojó como resultado la existencia de un conjunto de sistemas CRM ante la decisión de desarrollar uno desde cero. Se considera importante valorar la reutilización de una de las estudiadas pues ahorraría un valioso tiempo y esfuerzo de desarrollo, siendo OpenERP la más conveniente por las siguientes razones.

Este sistema brinda flexibilidad y simplicidad a la hora de realizar modificaciones y adaptaciones, logrando poder adaptarlo a las necesidades. Brinda integración con herramientas de negocios, utiliza un flujo de trabajo flexible y dinámico, pudiéndose agregar funciones y módulos e integrarlos a los ya existentes. Soporta plataformas heterogéneas, permite el uso del marco de trabajo OpenObject que posibilita que toda la lógica de negocio de OpenERP sea totalmente del lado del servidor, el cliente solo se tiene que encargarse de pedir los datos y este dinámicamente los muestra en la vista.

Debido que el sistema a utilizar es OpenERP y la necesidad de interactuar con la

información en tiempo real, se escoge como librería para la interacción con mensajería instantánea **XMPPPY**, que está desarrollada en python, siendo compatible con OpenERP. Entre los servidores de mensajería instantánea se decide el uso de **OpenFire** que cuenta con todas las características de los demás propuestos previamente y además incluye la administración web que facilita la configuración de forma sencilla.

Como herramienta de BI se utilizará **Pentaho**, debido a su capacidad a la hora de realizar las transformaciones deseadas, todos los programas desarrollados por Pentaho son creados y diseñados con independencia del resto y en esquema modular y su código es libre. Para la descripción de las vistas arquitectónicas se utilizará el modelo arquitectónico propuesto por CEIGE, el cual facilita una descripción de cada vista de forma detallada.

Capítulo 2 Propuesta de Solución

2.1 Introducción

En el presente capítulo se describirá la propuesta de solución a partir del uso del modelo arquitectónico del CEIGE el cual propone diferentes vistas arquitectónicas. La descripción de las vistas se llevará a cabo para obtener la línea base de la arquitectura y a partir de esta evaluar si la arquitectura propuesta es válida.

2.2 Vista de procesos

Un proceso es una secuencia ordenada de actividades interrelacionadas entre sí, que transforman las entradas en resultados, para dar respuesta o prestar un servicio al cliente, usuario o beneficiario, creando un valor intrínseco para los mismos (Victor, 2010). Estos procesos de negocio son la base para comprender mejor la forma en que opera un negocio en sus diferentes áreas. A continuación se brindará una explicación relacionada con los procesos que se han identificado para el desarrollo del sistema, que incluye el estudio de los procesos de OpenERP y de empresas interesadas en el negocio.

2.2.1 Modelo de procesos de la arquitectura

El sistema a desarrollar, cuenta con una serie de macro procesos tales como: **Mercadotecnia** es el encargado de realizar un estudio del mercado, se ejecutarán actividades que ayudan a las empresas a gestionar las relaciones con los clientes. **Compra/Venta** macro proceso encargado de registrar las solicitudes de compra, los pedidos de compras, las solicitudes de presupuestos, los pedidos de ventas y oportunidades. **Servicios** macro proceso que es donde se ofrecen prestaciones a los clientes como devoluciones, reparaciones, quejas, reclamaciones, para así lograr la satisfacción de los mismos. Cada macro proceso de estos descritos, registran una serie de procesos.

Mercadotecnia registra:

Análisis del mercado tiene como objetivo determinar las posibilidades que existen de transformar las ventas potenciales en ventas reales, y conocer si la inversión a realizar es rentable. Este proceso tiene como subprocesos: Definición del mercado,

Consideración del entorno y Apreciación de clientes potenciales. (Ver **Anexo *1***).

Selección del mercado objetivo se realiza una estimación del tamaño total del mercado, su crecimiento y rentabilidad, siendo aportes claves para decidir en qué mercados y en qué nuevos productos hay que concentrarse. El proceso inicia cuando el director de mercadotecnia orienta la selección de un mercado objetivo. El especialista define los elementos de la investigación, realiza un cronograma de trabajo, identifica las posibles fuentes de información, procesa y analiza la información creando así una descripción del mercado. Luego el especialista de mercadotecnia elegirá las variables de segmentación de mercado permitiéndole identificar los distintos segmentos y por último elegir los segmentos objetivos. (Ver **Anexo *5***).

Diseño de estrategias de mercadotecnia luego de estudiar toda la información obtenida con la investigación de mercados, llega el momento de tomar decisiones estratégicas que permitan direccionarse, diferenciarse y posicionarse en el mercado. El proceso diseño de estrategias inicia cuando el director de mercadotecnia presenta a los asesores de cada centro el modelo a utilizar para elaborar la ficha comercial, explicando cada uno de sus componentes. El especialista de mercadotecnia crea la ficha comercial, en caso de que ya esté elaborada solo la actualiza para posteriormente analizar la información obtenida y enviarla al director de mercadotecnia para su aprobación. (Ver **Anexo *7***).

Definir planes de mercadotecnia tiene como objetivo precisar las acciones para ejecutar las estrategias y alcanzar el objetivo propuesto. Primeramente el director de mercadotecnia orienta la planeación de programas de mercadotecnia, para ello elabora un plan estratégico. Plan que luego será convertido en planeación táctica por el especialista de mercadotecnia y que el gerente de operaciones convertirá en planeación operacional. Luego el director definirá en un informe los planes de mercadotecnia y por último se elaborará un cronograma de trabajo para aplicar los planes definidos anteriormente. (Ver **Anexo *9***).

Ejecución del plan mercadotecnia es la etapa en el proceso de la mercadotecnia en el que se aplican los planes estratégicos y tácticos. Es el momento cuando se tiene que: producir o conceptualizar el producto o servicio destinado a satisfacer las necesidades y deseos del mercado objetivo, aplicar las políticas de precio que el mercado objetivo pueda y esté dispuesto a pagar, implementar los canales de distribución mediante los cuales el producto o servicio esté disponible en el lugar y

momento adecuado y promocionar o promover el producto o servicio con el objetivo de informar, persuadir y/o recordar al mercado objetivo sus beneficios y su disponibilidad en el mercado. (Ver **Anexo *11***).

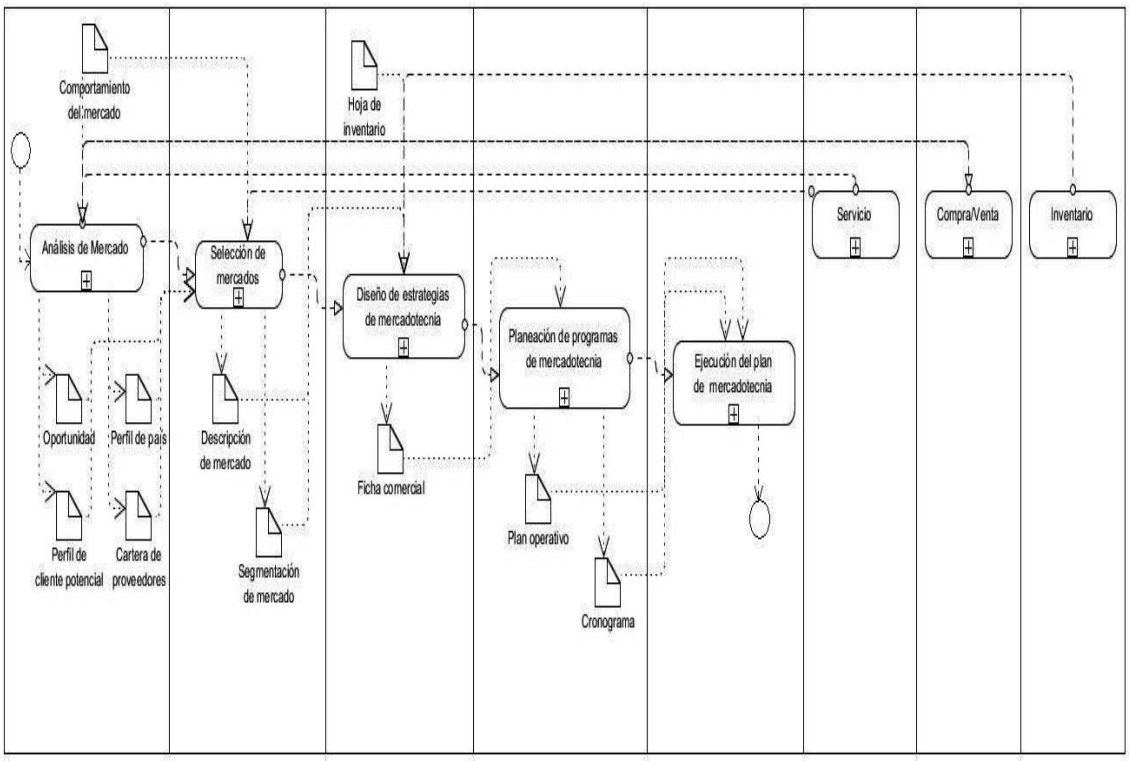


Figura 2 Descripción del macro proceso Mercadotecnia.

Compra/Venta registra:

Compra, primeramente se deberá detectar las oportunidades de compra, generándose a continuación la solicitud de compra y evaluándose el presupuesto para la realización de la compra. Una vez analizado el presupuesto se somete a aprobación la solicitud de compra. Siendo aprobada la solicitud, se le entrega al proveedor, se realiza la contratación, se establecen las obligaciones de pago y se realiza la compra. Realizada la operación, se factura la compra, se archiva el documento y se notifica la recepción realizada. (Ver **Anexo *13***).

Venta, primeramente se deberán detectar las oportunidades de ventas que existan, una vez concluida esta identificación se realiza el pedido de ventas, formalizando el pedido. Se analiza el pedido para la realización de la venta, aprobando el pedido y enviándolo. A continuación se realiza la contratación para establecer obligaciones de pago, se realiza la venta, la factura de venta, se actualiza el inventario y se notifica la

actualización. (Ver **Anexo *13***).

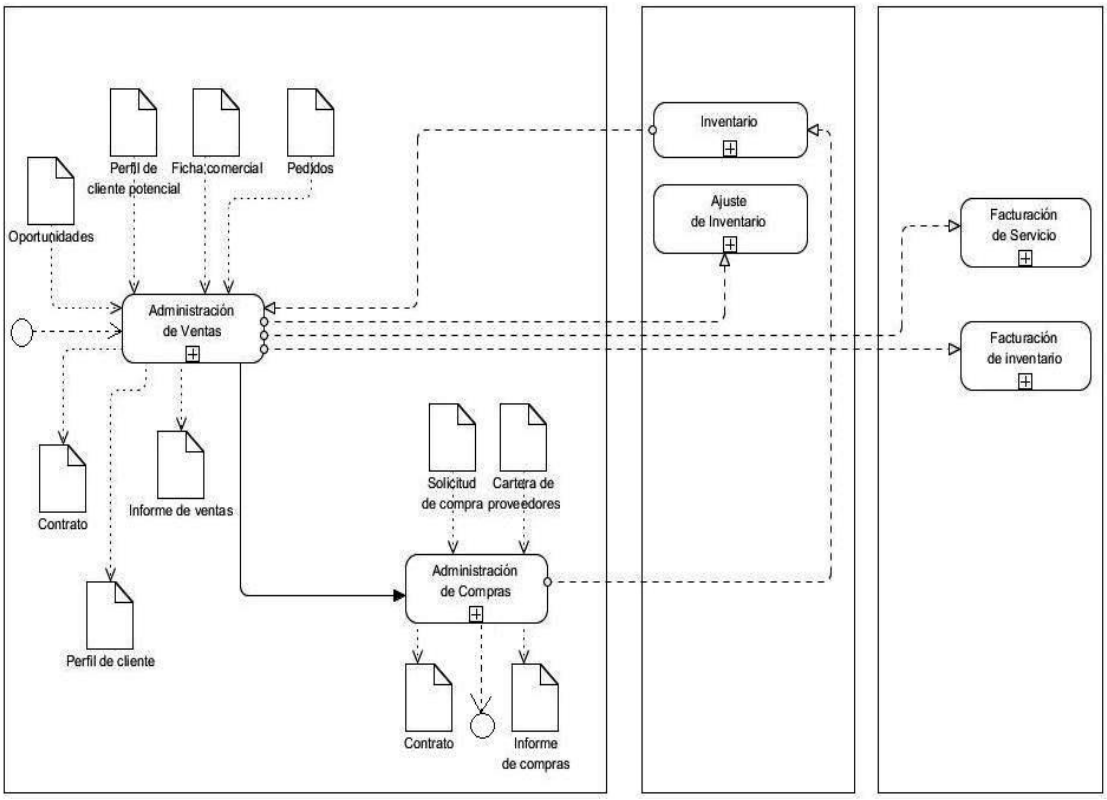


Figura 3 Diagrama del macro proceso Compra/Venta*.*

Servicios registra:

Atención al cliente, tiene como objetivo coordinar y ejecutar las acciones correctivas y preventivas para satisfacer las inquietudes presentadas por los clientes. Este proceso tiene como subprocesos: Gestión de solicitudes, Información de bienes y servicios. (Ver **Anexo *16***).

Medición de la calidad percibida, tiene como objetivo evaluar la calidad de los productos y servicios ofertados por la empresa y diseñar los métodos adecuados que posibiliten la obtención de los datos necesarios para ello. Este proceso tiene como subprocesos: Grado de satisfacción del cliente y Grado de respuesta a reclamos. (Ver **Anexo 19**).

Soporte, tiene como objetivo disminuir el tiempo de espera por una reparación o devolución, resolver las inquietudes de los clientes sobre la operación y mantenimiento de los productos ofertados por la empresa, además de brindar capacitaciones a los

clientes de la empresa. Este proceso tiene como subprocesos: Asistencia Técnica, Devoluciones y Reparaciones. (Ver **Anexo 21**).

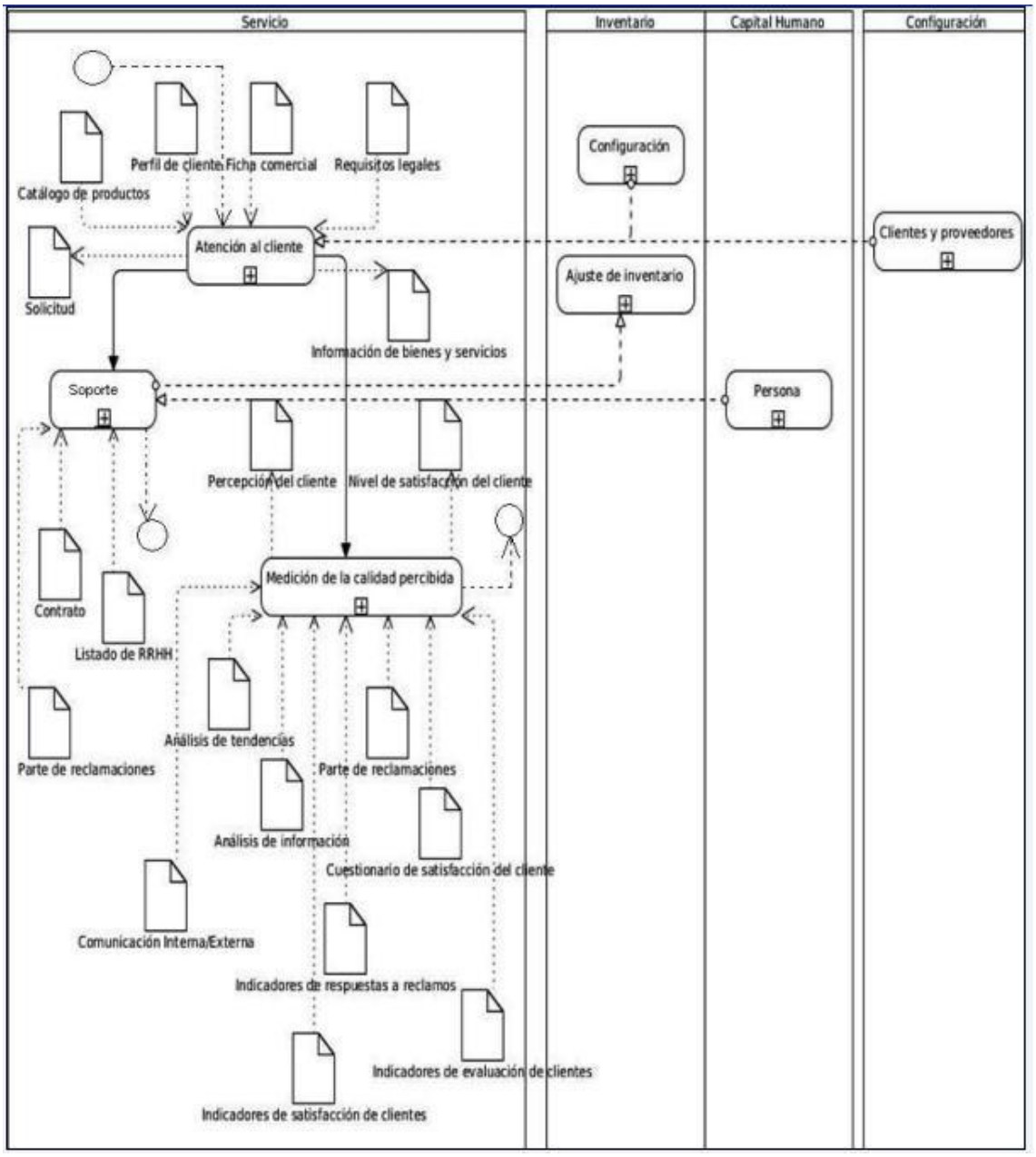


Figura 4 Diagrama del macro proceso Servicios.

La descripción de la presente vista, proporciona una visión global de los procesos que soportará la arquitectura. También facilitará definir los componentes, con los cuales contará el sistema, así como la dependencia que existente en ellos, percibiendo las entradas y salidas que tendrá, para el establecimiento de los servicios entre componentes.

2.3 Vista de presentación

A continuación se procede a la descripción de la vista de presentación, se enfoca en la apariencia del sistema. En OpenERP prevalece el color gris en todo el área del centro, en la parte superior izquierda se encuentra el logo del sistema de color rojo y negro, el menú de color rojo es el principal, donde se encuentran los módulos instalados, cada módulo es identificado por un botón de color rojo con su nombre. Además estos módulos se pueden encontrar en el cuerpo del sistema, identificado por un ícono que se asocie al módulo. Los botones que se encuentran en el menú principal cuando son seleccionados toman color gris. Las acciones asociadas a estos módulos cuando son seleccionados aparecen en la parte izquierda media en forma de árbol con un color gris más intenso. La interfaz visual de cada una de estas acciones es de color gris y los botones de las acciones también son de color gris. Toda la definición de colores es sobria y propicia la utilización del sistema por largos períodos de tiempo sin cansar al usuario. Es importante señalar que cada uno de los elementos visuales descritos son configurables, permitiendo su modificación y por consiguiente el uso de diferentes plantillas.

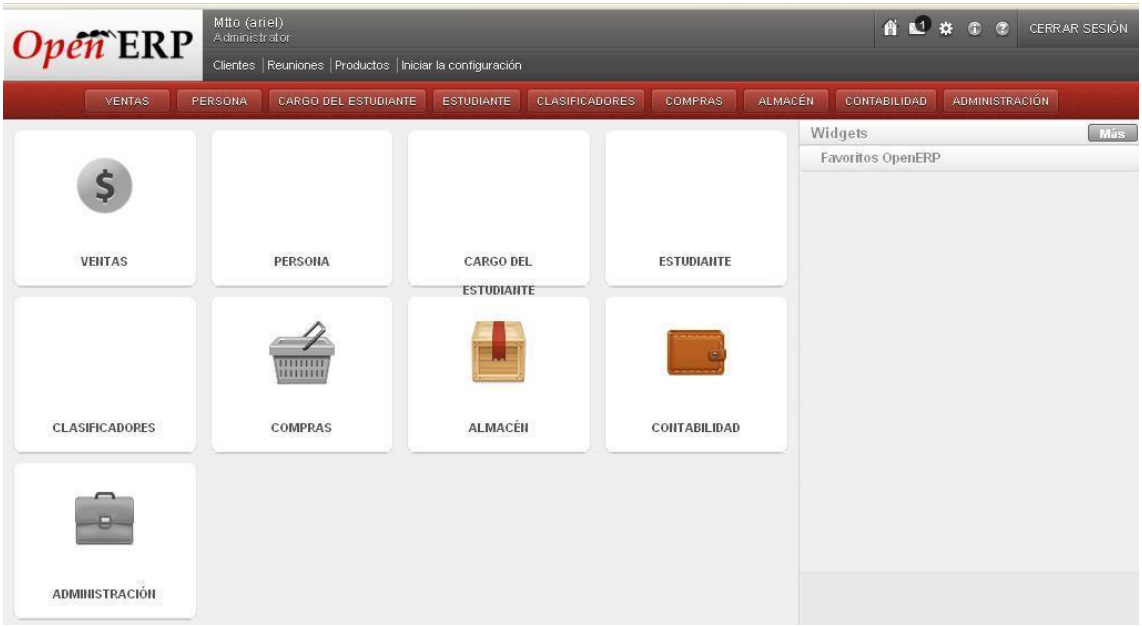


Figura 5 Vista de presentación.

Figura 6 Selección de un módulo.

2.4 Vista de sistema

Por las características que presenta el dominio de los negocios a incidir con el desarrollo de la aplicación, las tendencias y experiencias del desarrollo de otros sistemas CRM, se decide adoptar para el desarrollo horizontal del sistema el **estilo arquitectónico** orientado a componentes.

La **estructura de empaquetamiento** queda de la siguiente forma, el sistema quedaría constituido por un conjunto de componentes que responden a varias funcionalidades, un grupo de interacciones entre estos componentes respondiendo a las distintas integraciones y dependencias originadas en el negocio. Estos componentes están agrupados en una unidad mayor, denominada sistema, la cual responden a las áreas de procesos más generales identificadas en el negocio. El siguiente diagrama muestra algunos requisitos por componentes, el resto se encuentran en el artefacto descripción de requisitos generados por los analistas.

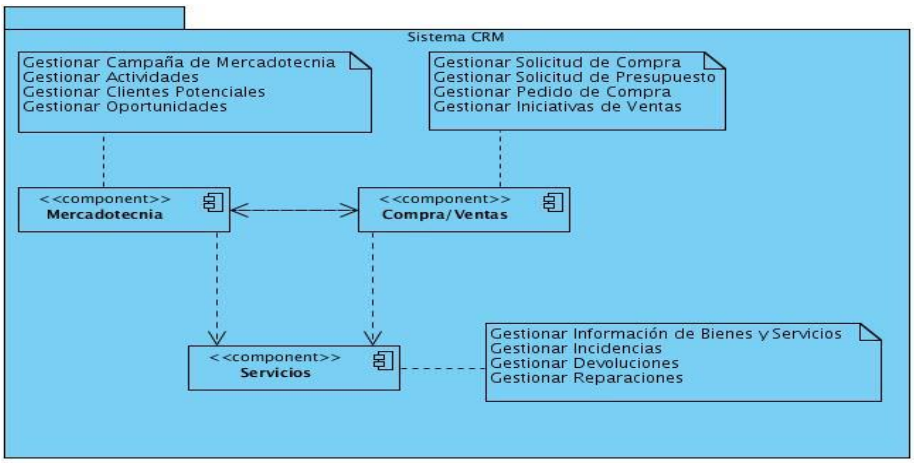


Figura 7 Estructura de empaquetamiento.

El **mapa general de componentes** es otro de los elementos a tener en cuenta en la descripción de la presente vista. Artefacto que contendrá todos los componentes definidos en el sistema, así como las distintas dependencias existentes entre los mismos, con el objetivo de obtener una vista general del sistema a partir de las dependencias e integraciones de los componentes. Permite evaluar cada uno de ellos según su complejidad así como, qué tan importante es desde el punto de vista de la integración.

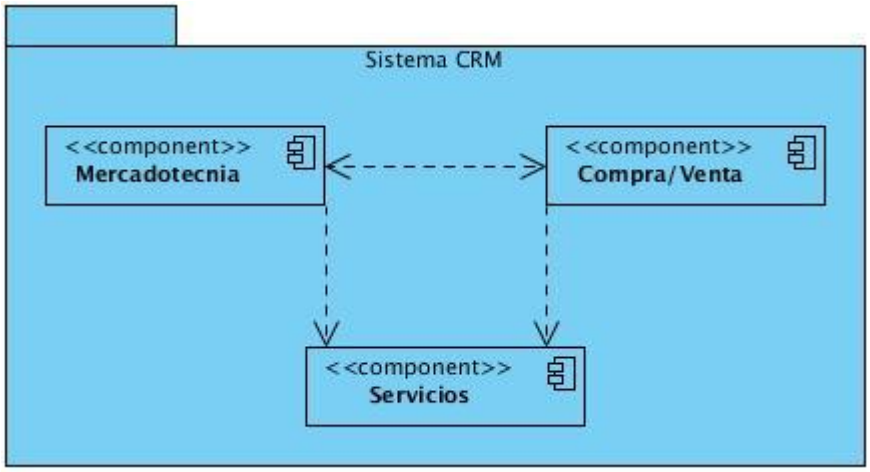


Figura 8 Mapa general de componentes.

La **especificación de componentes** es otro de los elementos, en el mismo se listan todos los componentes definidos en el sistema, y de cada uno de ellos se especificará el nombre, sus principales características, los servicios que provee con sus detalles. Para más información consultar el documento entregable especificación de componentes.

Para el desarrollo del sistema CRM se **reutilizará** el componente de Mercadotecnia, Compra/Venta y Servicios, se encuentran desarrollados en OpenERP su reutilización permitirá culminar el desarrollo en menor tiempo y con mayor calidad al ser componentes probados.

Los **estilos arquitectónicos** utilizados en el desarrollo de la aplicación son: estilo basado en componentes, se evidencia en la separación de cada uno de los macro procesos en componentes. También se utiliza el estilo MVC, OpenERP hace uso de este estilo, dado que cuenta con la clase controladora OSV, la vista se encuentra desarrollada en un XML y los datos son consultados mediante la clase entidad, la misma tiene que heredar de la clase controladora. Otro de los estilos que se utilizan es

el estilo orientado a objeto, en OpenERP un objeto A puede instanciar a un objeto B, después de haber especificado que el A depende del B, otro de los elementos que también se pueden realizar es la herencia entre objetos.

El artefacto **análisis de complejidad y criticidad** contiene todos los componentes definidos en el sistema, y por cada uno los requisitos funcionales que implementa, la complejidad de esos requisitos, y una serie de datos desde el punto de vista de integración. A partir de la dependencia entre componentes y la prioridad de requisitos se determinan la complejidad de los componentes y la criticidad de los mismos. Para más información revisar artefacto entregable análisis de complejidad y criticidad.

2.5 Vista de seguridad

El acceso a OpenERP es mediante usuario, grupos, roles, accesos de menú, controles de acceso.

Usuarios: Los usuarios del sistema de OpenERP acceden al sistema desde cualquiera de las opciones válidas de clientes disponibles. Los usuarios representan a las personas físicas.

Grupos: Los grupos determinan los derechos de acceso a los diferentes recursos.

Hay tres tipos de derechos:

- ✓ El acceso a la escritura: la creación.
- ✓ El acceso a la lectura: lectura de un archivo.
- ✓ La ejecución de acceso: los botones de flujos de trabajo o asistentes.

Roles: Permite distribuir roles dentro de los grupos y establecer usuarios con los roles definidos para ese grupo. Por ejemplo: Se establecieron los grupos mercadotecnia, servicios y compra/venta; a los dos primeros se decidió incluir el rol de administrador y vendedor, además al último el de secretario. Esto permite que solo los usuarios del grupo mercadotecnia y servicios puedan ser administradores.

Acceso al menú: Los usuarios, en función del rol que tengan asignado, estarán facultados para acceder a determinadas secciones del menú. Así por ejemplo al grupo de usuarios de servicios, se les puede establecer que no dispongan de acceso al menú del área de mercadotecnia.

Reglas: Es una forma que permite simplificar el proceso de otorgar a determinado roles a grupos de usuarios. Especialmente útil cuando se tiene múltiples usuarios.

Otro de los mecanismos para la seguridad en desarrollo del sistema CRM, es el uso de un certificado a la hora de consumir un servicio que se encuentre publicado del lado de Cedrux. Para obtener el certificado se hará uso de la función *obtenerCertificado*() a la cual se le pasa por parámetro el usuario y la contraseña, información que se mueve cifrada.

La forma de acceso a OpenERP, explicada anteriormente y el mecanismo de consumo de servicios posibilitan la accesibilidad, integridad y la confidencialidad de la información, proporcionando un sistema más confiable para el usuario final.

2.6 Vista tecnológica

Entorno de Desarrollo Tecnológico

El **sistema operativo** propuesto para el desarrollo es Ubuntu 11.04 o mayor. La tecnología de ambiente integrado al desarrollo (**IDE**) es Eclipse, dado que es una herramienta utilizada en el CEIGE la misma permite el completamiento de código python y la integración con herramientas de versionado. Para el modelado se usará Visual Paradigm, herramienta que propone el centro para el modelado de sistemas. Para el **versionado**, tanto para la documentación como para el código fuente se utilizará, la herramienta de control de versiones RapidSVN, propuesta por el centro y como **servidor de aplicaciones**, apache. Se hace necesario el uso de la máquina virtual de java, para el funcionamiento de Pentaho y Openfire dado que estas herramientas se encuentran desarrolladas sobre java.

OpenERP, utiliza el **marco de trabajo** OpenObject, consta con un proceso denominado desarrollo rápido de aplicaciones (RAD, siglas en inglés), acelerando el ciclo de desarrollo de un software, sus principales características son: la inclusión de un mapeo relacional de objetos (ORM, siglas en inglés) y de un motor de flujo de trabajo. OpenERP incluye el marco de trabajo Cherrypy para el desarrollo de la capa de presentación.

También contiene un motor de informes que permite la impresión de diferentes documentos, desde facturas hasta estadísticas. En OpenERP se tiene, por el momento, las siguientes formas de generar un informe:

- Informes personalizados, que son creados directamente desde la interfaz del cliente sin necesidad de programar nada. Estos informes son una representación directa de los objetos de negocio.
- Informes personalizados utilizando Openerport, estos informes se crean a partir de dos archivos, lenguaje de marcas extensible (XML, siglas en inglés), una plantilla que indica los datos disponibles en el informe y una hoja de estilo XSL: RML.
- Informes grabados en código.
- Plantillas de OpenOffice.org Writer.

Como lenguaje de programación para la lógica de negocio se usará python, Postgres como gestor de base de datos y XML en la capa de presentación. Este se basa en una arquitectura cliente/servidor, se comunican mediante el protocolo XML-RPC o NET-RPC.

Los **servicios** web ofrecidos del lado del servidor de OpenERP son: SOAP, XML-RPC, NET-RPC, Cedrux brinda los servicios web mediante WSDL. El **registro legal de la arquitectura tecnológica** es completamente de código abierto, basado en licencia GPL.

La descripción de la presente vista facilita la selección de las tecnologías necesarias para el posterior desarrollo del sistema CRM a desarrollar.

2.7 Vista de integración

El **registro de nodos de integración** del sistema CRM como se explicó anteriormente en la vista de procesos, cuenta con tres componentes principales, se integran mediante las relaciones entre clases, que permite realizar el marco de trabajo OpenObject (one2one, one2many, many2one, many2many). Las relaciones entre componentes quedan registradas en el entregable matriz de integración de componentes. La matriz entre componentes, en el eje vertical muestra los componentes que proveen servicios y en el eje horizontal los componentes que los consumen. Estos servicios se encuentran identificados por un código que permite su

búsqueda en el entregable “Especificación de Componentes”. Este último artefacto especifica la denominación, parámetros de entrada y salidas de los servicios. Ambos entregables contienen toda la información necesaria para que el programador o cualquiera de los desarrolladores que necesiten servicios conozcan todos sus detalles.

OpenERP brinda los datos a otro sistema, mediante el **estándar de integración** NET-RPC, este estándar de comunicación propicia los servicios web para lograr la integración entre sistemas, permitiéndole al cliente realizar procedimientos remotos, la llamada de funciones con sus parámetros. El resultado de las llamadas se transportan a través del protocolo HTTP y codificado en XML. Aunque no se han identificado sistemas externos al CRM, que necesiten del consumo de sus servicios este mecanismo permite que en un futuro esto sea posible, garantizando la escalabilidad de la arquitectura.

OpenERP consumirá la información de Cedrux a través del uso de la librería SOAPpy, es la encargada de consumir el WSDL publicado. Para el uso de esta librería se crea una clase llamada Service, en ella se implementa el decorador de python, se le pasa por parámetro una función y devuelve otra función, además se garantiza la seguridad estableciendo un mecanismo de certificación siempre antes del consumo de un servicio. Incluye SOAP como protocolo de intercambio de información. A continuación se muestran unas imágenes, la primera es la implementación de la clase Service y la segunda la llamada a la clase Service en la controladora:

Service.py

```
# -*- coding: UTF-8 -*-
from SOAPpy import WSDL

def connect_wsdl(url, debug=False):
    def decorator(function):
        def wrapper(*args, **kwargs):
            proxy = WSDL.Proxy(url)
            if debug:
                proxy.soapproxy.config.dumpSOAPOut = 1
                proxy.soapproxy.config.dumpSOAPIn = 1
            return function(proxy, *args, **kwargs)
        return wrapper
    return decorator
service = @connect_wsdl('http://localhost:5901/mantenimiento/webservices/MantenimientoSoapService.wsdl',debug=True)
```

Figura 9 Implementación de la clase Service.

module_unidades.py

```
# -*- coding: UTF-8 -*-
from osv import osv
from osv import fields
from Service import Service

class module_unidades(osv.osv):
    _name = "module_unidades.type"
    _description = "Tipo de Unidad"
    _columns = {
        'name': fields.char('Nombre', size=64, required=True),
        }

@service
def get_tipos_unidades(self, cr, uid, integrator, context):
    datos = integrator.tiposUnidades(11000000001)
    for unidades in datos:
        vals = {
            'name': unidades['denom']
        }
        self.create(cr, uid, vals, context)
module_unidades()
```

Figura 10 Forma de consumir servicios en OpenERP.

Uno de los artefactos de mayor peso en esta vista, es la **matriz de integración de sistemas**. Esta matriz tiene la misma estructura que la matriz entre componentes, se documentan los servicios que necesita el CRM de Cedrux para lograr su

funcionamiento, consulta documento entregable, matriz de integración entre sistemas. A continuación se mostrará una vista general de despliegue del sistema donde se podrá observar lo explicado en esta vista.

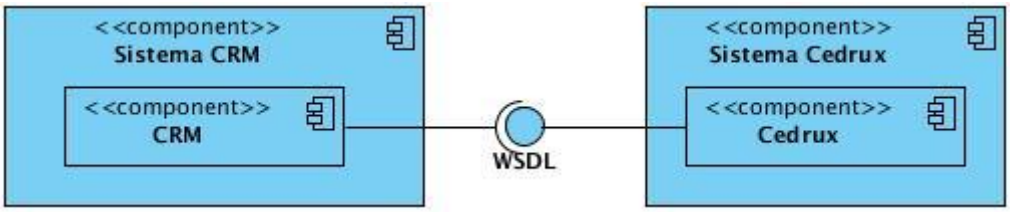


Figura 11 Vista de integración entre sistemas.

La descripción de la presente vista facilita entender con mayor claridad la forma en que se integra OpenERP con Cedrux, los estándares de integración que usa, junto con el protocolo de comunicación. La vista también responde al cumplimiento de los objetivos planteados, debido a que da soporte y demuestra la posibilidad de escalar el CRM para brindar en un futuro servicios, consumirlos de Cedrux o cualquier otro sistema que implemente servicios web con WSDL. Además provee mecanismos de seguridad con la implementación del patrón de diseño decorador para establecer un certificado de autenticidad previo al consumo de servicios. También se realiza una explicación del uso de la librería SOAPpy que es la encargada de consumir los servicios de Cedrux.

2.8 Vista de datos

En el presente epígrafe se realizará la descripción de la vista de datos. El ORM de OpenERP facilita que una vez que se acceda a la aplicación se cree la base de datos con la que se trabajará. La base de datos creada estará organizada en un solo esquema el cual contendrá todas las tablas del sistema. Para persistir cada uno de los objetos en la base de datos se debe crear una clase entidad, una vez creada esta clase con sus atributos se procede a cargar la entidad creada, este proceso se realiza en el módulo de administración, en el mismo existe una opción con el nombre aplicar actualizaciones programadas, al ser seleccionada se crea físicamente la tabla en la base de datos.

Open ERP es capaz de llevar del modelo entidad relación al modelo relacional, esto quiere decir que crea las tablas y sus relaciones a partir de las clases entidad y sus relaciones. Si las clases entidad con sus relaciones no presentan atributos

compuestos o derivados y además no hay dependencias parciales entre ellos y sus claves, cuando se apliquen las actualizaciones programadas, se tiene la garantía de que el modelo físico se crea en tercera forma normal pues elimina la dependencia funcional transitiva entre los atributos que no son claves. Cada una de las entidades que se creen, es a partir del modelo entidad relación que es creado por el arquitecto de datos, el cual debe de garantizar que se encuentre normalizado. A continuación se muestra un ejemplo de cómo crear una entidad.

module_estudiante.py

```
# -*- coding: UTF-8 -*-
from osv import osv
from osv import fields

class module_estudiante_tipo(osv.osv):
    _name = "module_estudiante.type"
    _inherit = "module_persona.type"
    _description = "Estudiante"
    _columns = {
        'note': fields.char('Nota', size=64, required=True,help='Nota del Estudiante en una Asignatura'),
        'asig': fields.char('Asignatura', size=64, required=True,help='Asignatura en la que se evaluo'),
        'cargo':fields.many2one('module_cargo_estudiante.type','Cargo',select = True,help='Cargo del Estudiante')

    }
module_estudiante_tipo()
```

Figura 12 Crear una entidad.

Para la realización del modelo de datos se utilizó como patrón de diseño llave subrogada que implementa el ORM de OpenERP, ya que se decide generar una llave única para cada entidad, en vez de usar un atributo identificador en el contexto dado. La aplicación del mencionado patrón permite que las tablas sean más fáciles de consultar por el identificador dado, ya que se conoce que tiene el mismo tipo en cada tabla. A continuación se muestra el modelo de datos de cada uno de los componentes a desarrollar.

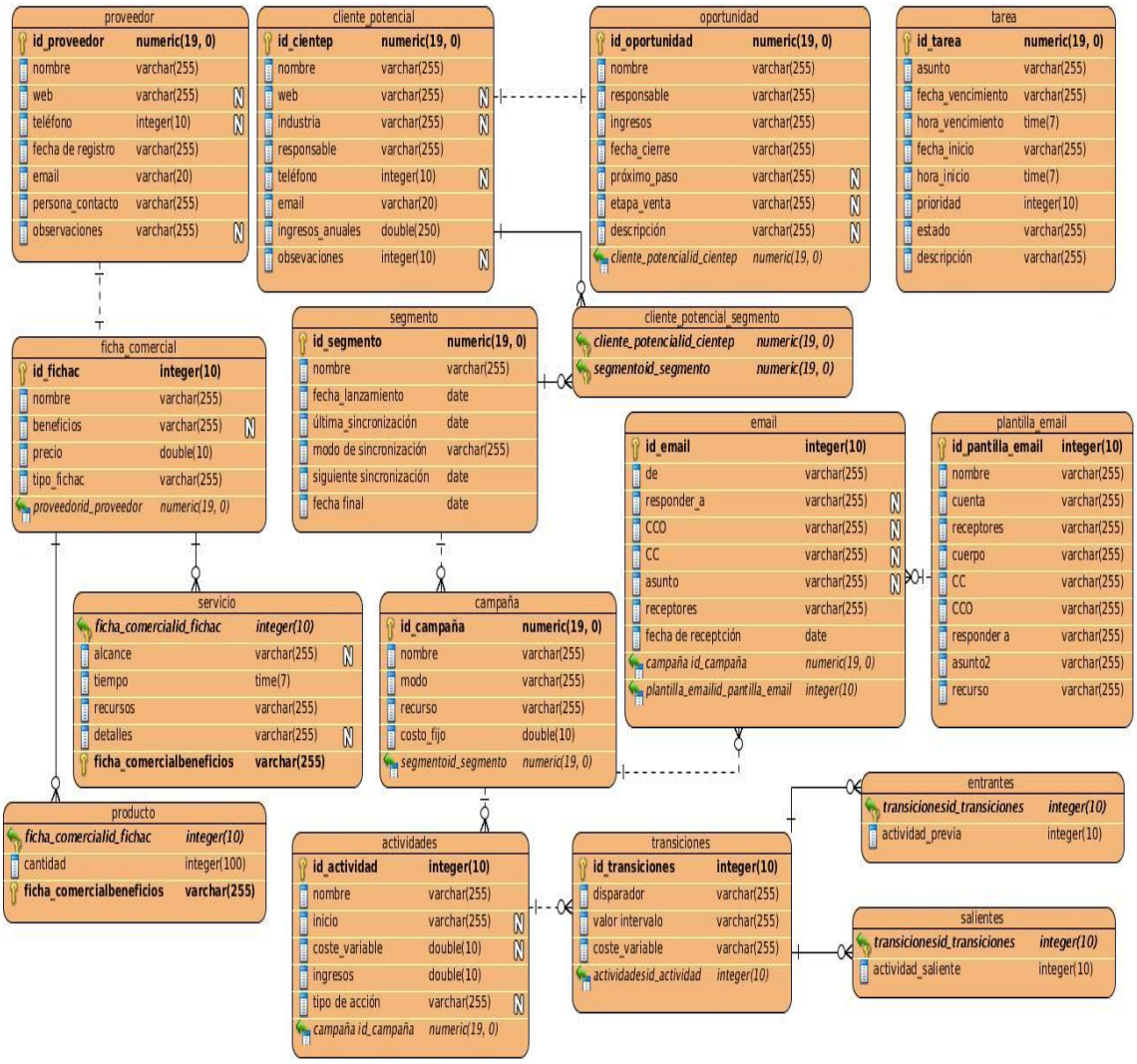


Figura 13 Modelo entidad relación de Mercadotecnia.

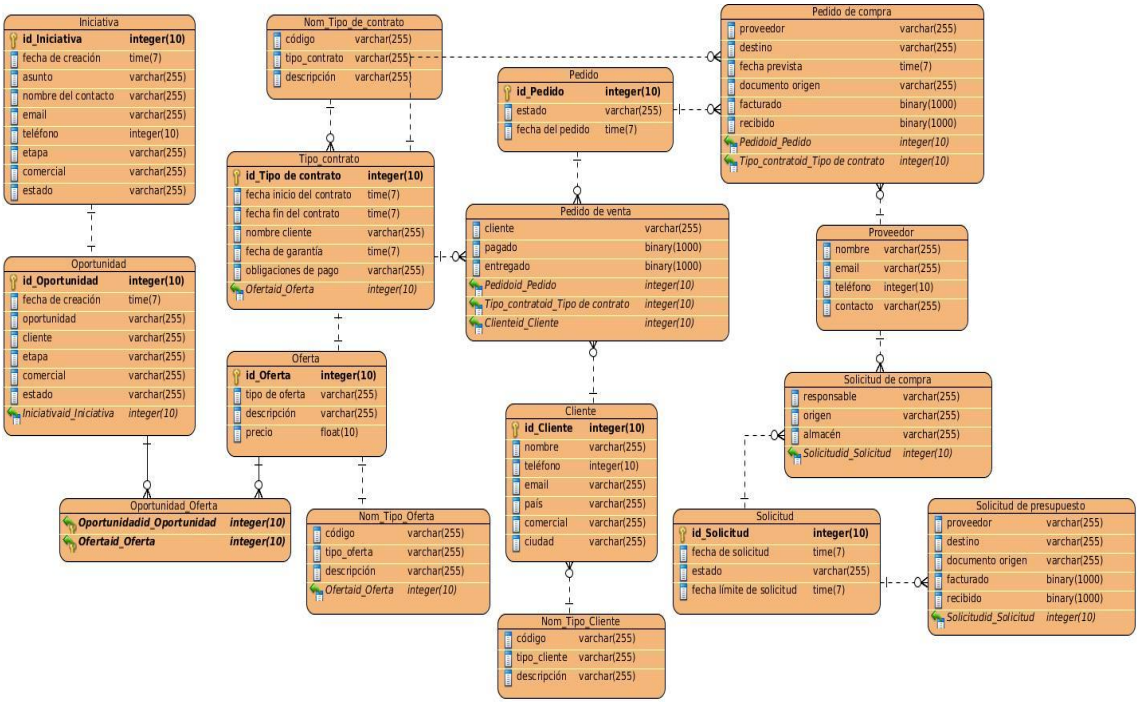


Figura 14 Modelo entidad relación de Compra/Venta.

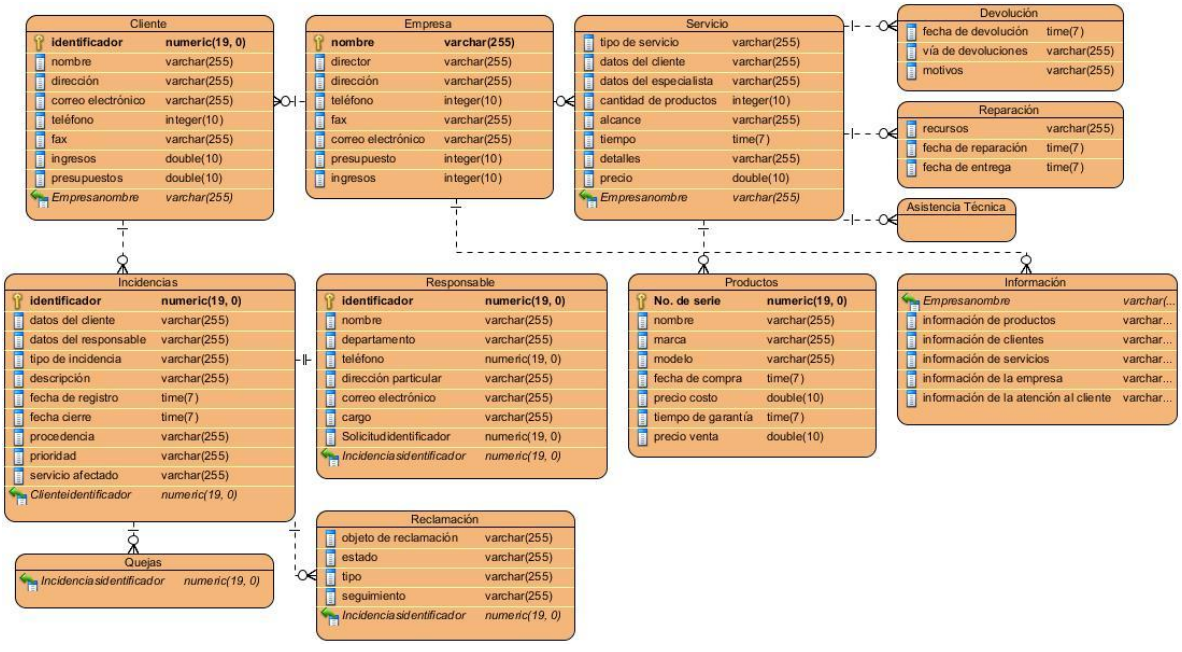


Figura 15 Modelo entidad relación de Servicios.

2.9 Vista de despliegue e infraestructura

A continuación se especificará la **arquitectura de infraestructura**, en la misma los nodos con capacidad de procesamiento son: dos servidores de aplicaciones y un

servidor de base de datos. A continuación se muestra el diagrama de despliegue de configuración del software:

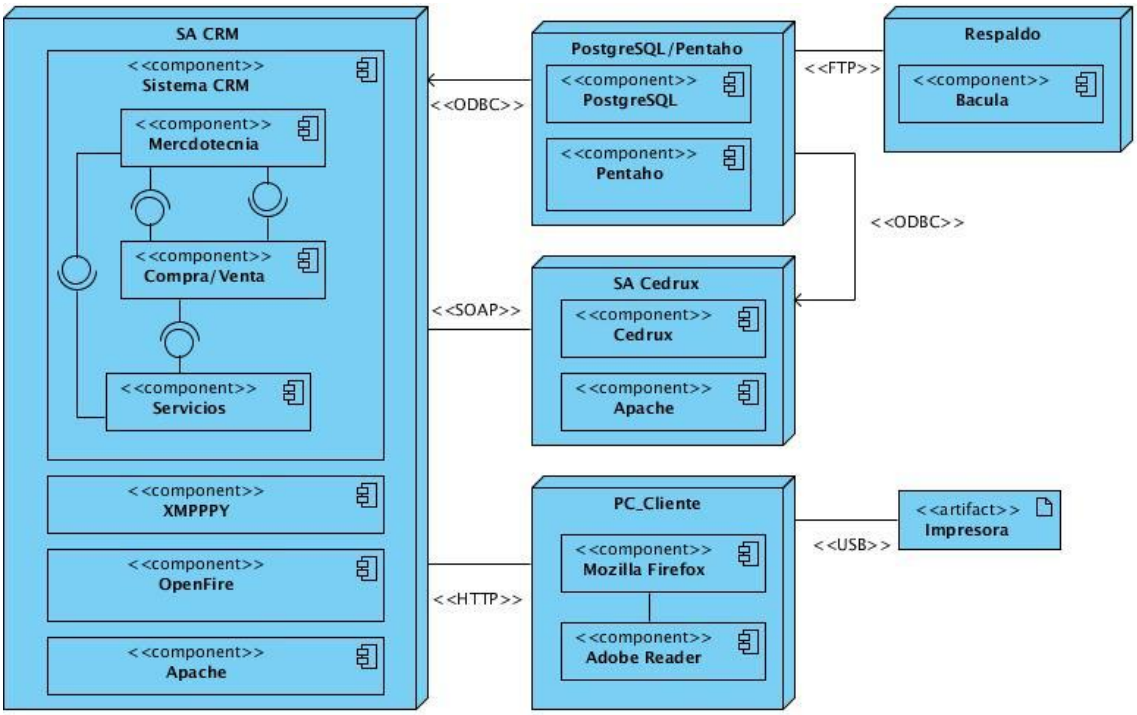


Figura 16 Diagrama de despliegue de la configuración del software.

Descripción de elementos e interfaces de comunicación

<<Nombre tipo de conexión>>: Características físicas de la conexión

<<HTTP>>: El protocolo de transferencia de hipertexto es un protocolo que define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones, sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor, además opera en la capa de presentación (W3C, 2011).

<<FTP>>: El protocolo de transferencia de archivos es utilizado para la transferencia de archivos entre sistemas conectados a una red, basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo (W3C, 2011).

<<ODBC>>: Es un estándar de acceso a bases de datos, el objetivo de ODBC acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS, siglas en inglés) almacene los datos. ODBC logra esto al insertar una capa intermedia denominada nivel de interfaz de cliente SQL, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda (FileMaker, 2010).

Descripción de los nodos

PC_Cliente: Brinda la interfaz para que el usuario intercambie con el sistema. Se conecta con el servidor de aplicaciones del sistema CRM a través del navegador usando el protocolo HTTP.

Servidor de Aplicaciones: Es el encargado de atender las solicitudes del nodo PC_Cliente, además puede establecer la comunicación con otro nodo servidor de aplicación a través del protocolo SOAP para dar respuesta a las solicitudes. El servidor debe contar con altas prestaciones dado que puede estar realizando diferentes peticiones a la vez.

Servidor de BD: Responde a las peticiones de almacenamiento y recuperación de datos solicitada por los servidores de aplicación a través del protocolo ODBC.

Visto la descripción relacionada con la descripción de los nodos se expondrán las prestaciones de hardware y software para el correcto funcionamiento del sistema.

Especificación de los elementos de despliegue

Requerimientos de Hardware

Servidor de aplicación/PC_Cliente	Requerimientos Mínimos	Requerimientos Máximos
Servidor de Aplicación Web	**Procesador**: 3.00 GHZ **RAM**: 1GB **Disco duro**: 40 GB **UPS**: 1 **Lector de CD**: 1 **Tarjeta de Red**: 1	**Procesador**: Dual Core. **RAM**: 4GB **Disco duro**: 40 GB **UPS**: 1 **Lector de CD**: 1 **Tarjeta de Red**: 1

Servidor de Base de Datos	**Procesador**:3.00 GHZ **RAM**: 1GB **Disco duro**: 40 GB **UPS**: 1 **Lector de CD**: 1 **Tarjeta de Red**: 1	**Procesador**: Dual Core **RAM**: 4GB **Disco duro**: 80 GB **UPS**: 1 **Lector de CD**: 1 **Tarjeta de Red**: 1
PC_Cliente	**Procesador**: 1.40 GHZ **RAM**: 256 MB **Tarjeta de Red**: 1	**Procesador**: 3.00 GHZ **RAM**: 1 GB **Tarjeta de Red**: 1

Requerimientos de Software

Servidor	Especificaciones
Servidor de Aplicación Web	**Sistema Operativo**: Windows Server 2003 o superior, Debian Server 5.0 o superior, Ubuntu Server 11.04. **Servidor de Aplicaciones**: Apache 2.2.4 **Librerías adicionales**: Python 2.7
Servidor de Base de Datos	**Sistema Operativo**: Windows XP o superior, Debian 5.0 o superior, Ubuntu 11.04 o superior. **Gestor de Base de Datos**: PostgreSQL 8.2 o superior. **Herramienta de BI**: Pentaho 1.7 o superior
PC_Cliente	**Sistema Operativo**: Windows XP o superior, Debian 5.0 o superior, Ubuntu 11.04 o superior. **Navegador Web**: Internet Explorer 7.0 o superior, Mozilla Firefox 12.0 o superior, Google Chrome 9.0 o superior, Opera 10.0 o superior. **Visualizador de ficheros PDF**: Adobe Reader 9.0

Mecanismos de salva y recuperación ante fallos

Para el mecanismo de salva y recuperación ante fallos, se utiliza el sistema Bacula, como colección de herramientas de respaldo, para permitir la copia y restauración de ficheros dañados o perdidos (Bacula, 2012). El proceso de copia se realiza de forma automática hacia otra máquina independiente donde normalmente se almacenan las salvas. El proceso de salvas de la información estará programado para que se realice diariamente. Una vez corregidas las fallas del servidor de base de datos, se recupera la información desde el servidor de respaldo hasta el de base de datos, mediante el sistema Bacula.

Esta vista permite el estableciendo de mecanismos de seguridad para el respaldo de los datos. Incluye las funciones de todos los nodos del sistema y los protocolos de comunicación para lograr la integración. Permite la valoración de nuevo equipamiento si se desea incrementar o escalar el sistema.

2.10 Conclusiones parciales

Se puede expresar que se encuentran descritos los elementos imprescindibles para la construcción del sistema CRM, dado que se ha realizado una descripción de la **Línea Base de la Arquitectura**, a través de sus vistas arquitectónicas dándole cumplimiento a uno de los objetivos más importantes de la investigación. En esta descripción se ha podido presenciar las potencialidades con las cuales cuenta el sistema seleccionado.

Capítulo 3 Pruebas

3.1 Introducción

En el presente capítulo se analiza la propuesta de solución desde el punto de vista de la calidad, para evaluar cómo se le da cumplimiento al problema del trabajo de diploma y verificar si la arquitectura propuesta cumple con las tres variables planteadas en el problema.

3.2 ¿Por qué evaluar la arquitectura de software?

La arquitectura es el primer artefacto del ciclo de vida del desarrollo de un software, que incorpora importantes decisiones de diseño las son fáciles de tomar, pero difíciles de cambiar una vez que el sistema se aplica (Camacho, et al., 2004).

Por tal motivo, es de gran importancia realizar una correcta evaluación de la AS de un sistema, y esto se logra, mediante el uso de modelos de evaluación arquitectónicos que permiten obtener predicciones acerca de estos atributos de calidad (Camacho, et al., 2004).

Evaluar una AS sirve para prevenir los posibles desastres de un diseño que no cumpla con los requerimientos de calidad, para determinar qué tan adecuada es la AS de un sistema y para obtener información sobre donde está el riesgo, es decir fortalezas y debilidades identificadas en dicha AS (Camacho, et al., 2004).

Después de una evaluación, se pueden tomar algunas decisiones como: si se puede seguir el proyecto con las áreas de debilidad dadas en la evaluación, si hay que reforzar la AS o comenzar de nuevo toda la arquitectura.

3.3 ¿Cómo evaluar la arquitectura de software?

Existen diferentes técnicas utilizadas para la evaluación de atributos de calidad que requieren grandes esfuerzos por parte del ingeniero de software, para crear especificaciones y predicciones. Estas técnicas requieren información del sistema a desarrollar que no está disponible durante el diseño arquitectónico, sino al principio del diseño detallado del sistema (Camacho, et al., 2004).

En vista de que el interés es tomar decisiones de tipo arquitectónico en las fases tempranas del desarrollo, son necesarias técnicas que requieran poca información detallada y puedan conducir a resultados relativamente precisos. Las técnicas existentes en la actualidad para evaluar arquitecturas permiten hacer una evaluación cuantitativa sobre los atributos de calidad a nivel arquitectónico. Bosch propone diferentes técnicas de evaluación de la AS: evaluación basada en escenarios, evaluación basada en simulación, evaluación basada en modelos matemáticos y evaluación basada en experiencia (Camacho, et al., 2004).

3.4 Método para el análisis de las arquitecturas de software (SAAM)

SAAM (siglas en inglés) fue el primer método ampliamente promulgado y documentado creado originalmente para el análisis de la modificabilidad de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad, tales como: modificabilidad, portabilidad, escalabilidad e integrabilidad.

El método de evaluación SAAM, se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro, como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada (Kazman, et al., 2009). Estos escenarios fueron identificados a través de las variables planteadas en el problema y cubriendo los atributos de calidad desde las perspectivas de los usuarios finales, arquitectos de software, desarrolladores y administradores de los sistemas.

El presente método facilita una evaluación de la arquitectura de forma sencilla, a partir de seis pasos, estos son: desarrollo de escenarios, describir la arquitectura, clasificar y priorizar los escenarios, individualmente evaluar escenarios indirectos, evaluar la interacción escenario y creación de una evaluación global. Para poder describir estos es necesario tener la descripción de las vistas arquitectónicas y los escenarios a probar.

3.5 Procesos de pruebas a la solución.

SAAM define un conjunto de pasos bajo los cuales se enmarca el desarrollo de la metodología de evaluación, a continuación se describen los resultados de la aplicación del proceso.

Desarrollo de escenarios

A continuación se describirán los escenarios de la solución. Como sistema para lograr la integración se utilizará el Sistema de Gestión de Mantenimiento Vehicular (MTTOV), que tienen una arquitectura igual a la de Cedrux, debido que los dos sistemas se encuentran desarrollados con el mismo marco de trabajo.

Nombre del Escenario: Adicionar funciones y módulos.

- ✓ **Objetivo del negocio:** Se le debe informar al usuario que se logró instalar los módulos.
- ✓ **Descripción:** Los sistemas deben contar con una base tecnológica que permita ser escalable, se le agregarán tres módulos al sistema.
- ✓ **Ambiente:** Plataforma de OpenERP, permisos de escritura en la carpeta OpenERP-Server, permisos administrativos, existencia de los archivos __init_.py, __openerp_.py, la entidad y la vista.
- ✓ **Respuesta:** Se informa que se agregó el módulo satisfactoriamente o que ocurrió un error.
- ✓ **Medida de la Respuesta:** Instantánea.
- ✓ **Atributo de calidad que impacta:** Escalabilidad.

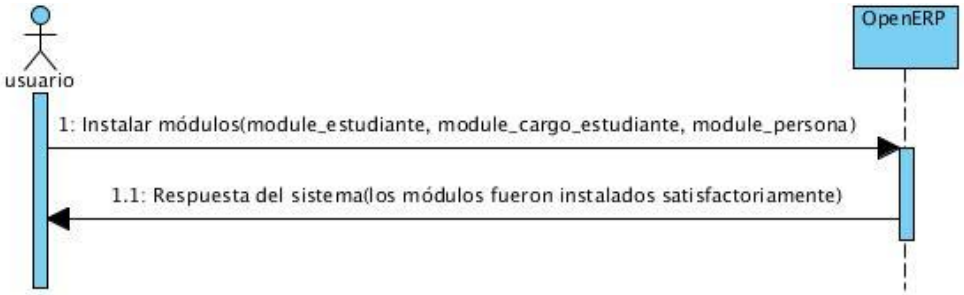


Figura 17 Descripción del escenario de escalabilidad.

Nombre del Escenario: Consumir servicios entre los sistemas OpenERP y MTTOV.

- ✓ **Objetivo del negocio:** Probar la interoperabilidad del sistema OpenERP con MTTOV.
- ✓ **Descripción:** Es la capacidad de un sistema, para poder integrarse con otro, sin importar el lenguaje de programación en el que se encuentre desarrollado. Un usuario invocará la llamada de un servicio que se encuentra publicado en el sistema MTTOV.
- ✓ **Ambiente:** Plataforma de OpenERP y MTTOV disponibles, el WSDL debe encontrarse público, debe de existir conexión de red, existencia de los archivos __init_.py, __openerp_.py, la entidad y la vista, permisos administrativos, importar clase Service en la entidad que consumirá información.
- ✓ **Respuesta:** Se muestran los resultados del servicio web.
- ✓ **Medida de la Respuesta:** Instantánea.
- ✓ **Atributo de calidad que impacta:** Interoperabilidad.

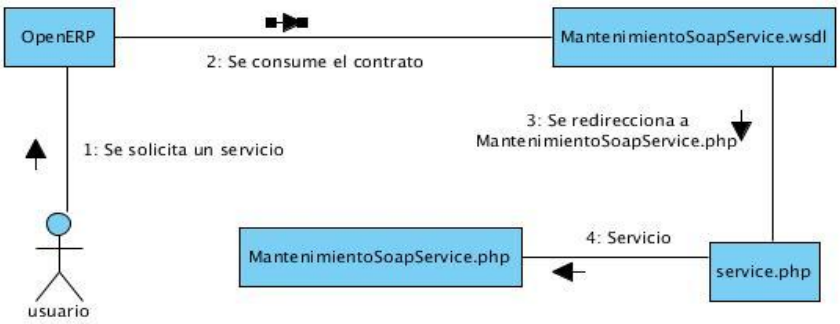


Figura 18 Descripción del escenario de interoperabilidad.

Nombre del Escenario: Bridar servicios entre los componentes del OpenERP.

- ✓ **Objetivo del negocio:** Probar la integración entre dos componentes.
- ✓ **Descripción:** Un componente necesitará información de otro.
- ✓ **Ambiente:** Plataforma de OpenERP, debe de existir la dependencia entre entidades, se debe de hacer uso de las relaciones entre clase de OpenERP, existencia de los archivos __init_.py, __openerp_.py, la entidad y la vista.
- ✓ **Respuesta:** Se muestran los resultados de la información solicitada.
- ✓ **Medida de la Respuesta:** Instantánea.

- ✓ **Atributo de calidad que impacta:** Interoperabilidad.

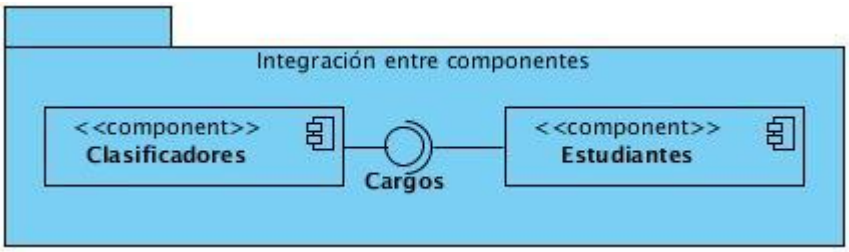


Figura 19 Pruebas de interoperabilidad entre componentes.

Nombre del Escenario: Verificar la disponibilidad, integridad y confidencialidad de la información a usuarios autorizados.

- ✓ **Objetivo del negocio:** Probar el nivel de seguridad del sistema.
- ✓ **Descripción**: Se requiere un sistema, que permita el acceso solo a aquellos usuarios autorizados. Un usuario intenta autenticarse con usuario y contraseña no valida.
- ✓ **Ambiente**: Tener grupos y roles definidos.
- ✓ **Respuesta**: Se muestra solo la información correcta y las funciones de acuerdo al nivel de acceso del usuario.
- ✓ **Medida de la Respuesta:** Instantánea.
- ✓ **Atributo de calidad que impacta**: Seguridad.

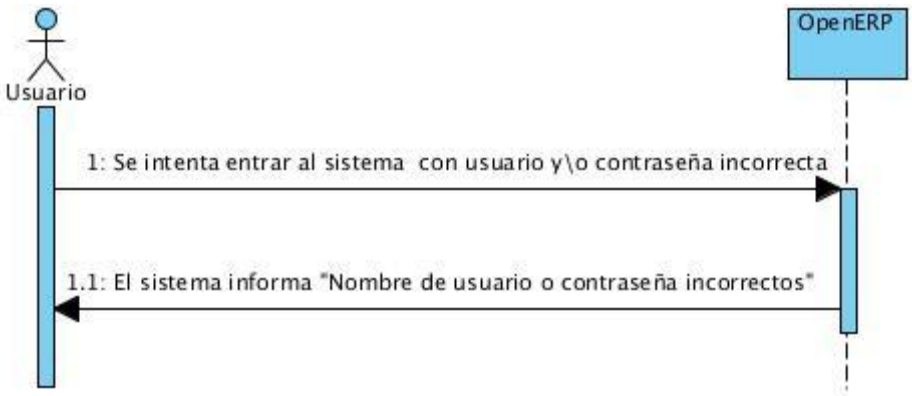


Figura 20 Descripción del escenario de seguridad.

Nombre del Escenario: Agregar librería para establecer comunicación entre sistemas.

Objetivo del negocio: Se debe probar si la plataforma posibilita la extensión.

- ✓ **Descripción**: Se requiere un sistema, que tenga la facilidad de adaptar el producto de software a los cambios que surjan durante el proceso de desarrollo. A la plataforma se le agregará, una nueva librería, para lograr consumir WSDL de MTTOV.
- ✓ **Ambiente:** Sistema MTTOV y la librería SOAPpy disponibles, el WSDL debe encontrarse público, debe de existir conexión de red, permisos administrativos, importar clase Service en la entidad que consumirá información.
- ✓ **Respuesta:** Lograr consumir los servicios utilizando la librería.
- ✓ **Medida de la Respuesta:** Instantánea.
- ✓ **Atributo de calidad que impacta**: Extensibilidad.

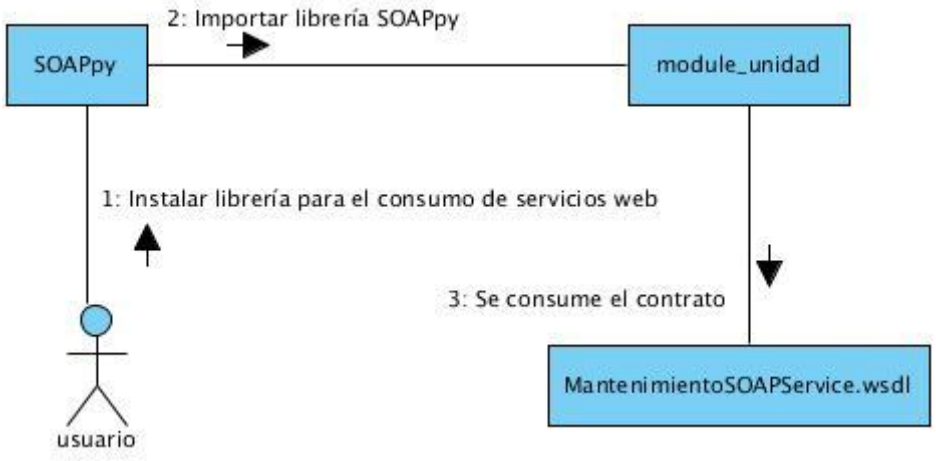


Figura 21Descripción del escenario extensibilidad.

Debido a las características específicas de la solución, se seleccionaron extensibilidad, seguridad, escalabilidad e interoperabilidad, para evaluar los atributos de calidad planteados en el problema.

Describir la arquitectura

En el capítulo 2 se definen los detalles arquitectónicos relevantes de la solución que han sido identificados y serán utilizados para la realización del proceso.

Clasificar y priorizar los escenarios

Tabla 4Priorización de los escenarios arquitectónicos.

Directos	Indirectos
Interoperabilidad	Extensibilidad
Escalabilidad	
Seguridad	

En este orden se considera que se encuentran priorizados los escenarios donde el más significativo es el primero y el menos significativo el último.

Individualmente evaluar escenarios indirectos

El escenario de extensibilidad, requiere agregarle a la plataforma de OpenERP la librería SOAPpy para comunicar los sistemas CRM y Cedrux, esta consumirá un WSDL de Cedrux. Es preciso explicar que la librería ya cuenta con un analizador de WSDL, que para consumir este fichero se debe de utilizar la clase WSDL.Proxy, que solo toma el fichero WSDL. La clase WSDL.Proxy expone las funciones disponibles como un diccionario de python. Desde el lado de Cedrux es necesario que se encuentre publicado el WSDL con el contrato establecido, para esto debe estar en el IoC externo la llamada a las funcionalidades requeridas para poder hacer uso de ellas.

Evaluar la interacción de escenario

Para este caso de prueba no existe más que un escenario indirecto por lo que carece de sentido el proceso de interacción entre escenarios que requieran modificaciones a la línea base trazada.

Creación de una evaluación global

Se considera que esta arquitectura cubre los escenarios más importantes directos y al menos está identificado el punto de extensibilidad para el escenario indirecto. Se identificaron cuatro escenarios de los cuales se consideraron tres directos y uno indirecto. En el caso del indirecto se estima que el costo para someter a la plataforma a estos cambios no tiene gran repercusión para el proyecto. No se identifican interacciones entre escenarios indirectos.

Pruebas de escalabilidad

La siguiente prueba se centra en la capacidad que posee el sistema para agrandar su diseño arquitectónico, se realizará el proceso de instalación de algunos módulos para observar qué sucede.

Proceso para instalar un módulo creado.

A continuación se muestran una serie de módulos que ya se encuentran instalados en OpenERP, se muestran en la parte superior de la aplicación. Para instalar uno nuevo se debe de ir al módulo de Administración y seleccionarlo. Al ejecutar la acción aparece un panel a la izquierda en el cual hay una opción con el nombre de Módulos, al seleccionar la opción Módulos, se abren una serie de opciones. Después se toma la opción importar módulo, este debe tener extensión .Zip que es la extensión que acepta OpenERP y se selecciona el botón importar módulo. Una vez realizado este proceso se va a la opción Módulos que se encuentra dentro de Módulos, se realiza la búsqueda del que se quiera instalar, se selecciona y se oprime el botón programar para instalación, después de realizado el proceso se va a la opción aplicar actualizaciones programadas, que se encuentra dentro de la opción Módulos y se selecciona el botón iniciar actualizaciones. Realizado todo el proceso explicado anteriormente se obtiene el siguiente resultado.

Figura 22 Resultado del proceso de instalación.

Pruebas de seguridad

Esta prueba se centra, en observar el nivel de seguridad de OpenERP, tratando de acceder al sistema con usuario y\o contraseña incorrecta, se mostrará la respuesta

que brinda el sistema. Para realizar el proceso primeramente se debe de seleccionar la base de datos, y a continuación se introduce usuario y contraseña. La figura 23 muestra una imagen tratándose de autenticarse un usuario que no pertenece al sistema.

Figura 23 Inicio de OpenERP.

Pruebas de interoperabilidad

En el presente acápite se realiza una explicación detallada de las pruebas de interoperabilidad, como se ha podido presenciar a lo largo del presente trabajo uno de los objetivos más significativos es la integración entre el sistema OpenERP y el sistema Cedrux. Para realizar las respectivas pruebas se propone la integración con el sistema MTTOV, producto que se encuentra desplegado y aprobado por CaliSoft teniendo la misma arquitectura del Cedrux.

Para realizar el proceso de integración entre los sistemas se parte de la explicación de los pasos a tener en cuenta, el proceso se realizará con el nomenclador Tipo de Unidades de MTTOV. Primeramente el usuario realiza una petición de los tipos de unidades, una vez realizada la invocación se realiza la llamada a la clase *module_tipo_unidad.py,* en esta clase es donde se importa la librería SOAPpy, la cual contiene una clase, WSDL.Proxy, pasándosele una dirección o sea el WSDL que está público en MTTOV. El WSDL se nombra *MantenimientoSoapService. WSDL,* en él hay publicado algunas funcionalidades del sistema MTTOV, se escoge para esta prueba tiposUnidades (). El contrato llamado realiza un direccionamiento a la clase *service.php,* es donde se crea el objeto SoapServer, pasándole como parámetro

WSDL, esta clase realiza la llamada a *MantenimientoSoapService.php,* se realiza una instancia del IoC externo de MTTOV, llamando a la funcionalidad listarTiposUnidades(), la funcionalidad está, en la clase *NomTipounidadService.php.* En este objeto se instancia la clase *NomTipounidadModel.php*, teniendo visibilidad a sus funcionalidades, en esta última clase es donde se realiza la llamada a la clase *NomTipounidad.php*, que es la encargada de realizar las consultas a la base de datos, y cargar los tipos de unidades que se encuentran registrados. A continuación se muestra un diagrama de secuencia.

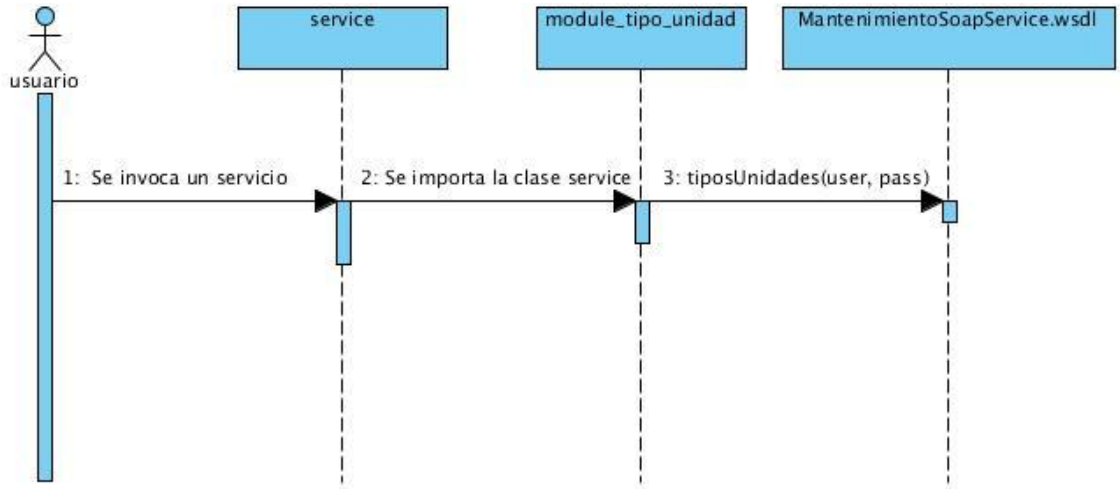


Figura 24 Diagrama de secuencia para la interoperabilidad.

Una vez realizado el proceso de interoperabilidad de los sistemas la respuesta es la siguiente:

Figura 25 Respuesta del sistema después de realizar la integración.

Otras de las pruebas de interoperabilidad, son las realizadas entre componentes. El proceso de integración entre componentes se realiza con el módulo de estudiantes y el módulo clasificadores, los cuales fueron ejemplos desarrollados por el grupo de trabajo del proyecto. Para realizar esta integración primeramente en el archivo __openerp__.py se debe de especificar la dependencia que existe entre los módulos, especificada esta dependencia se procede a realizar la llamada a la entidad que brindará la información. A continuación se muestra una imagen de cómo realizar la llamada.

module_estudiante.py

```
# -*- coding: UTF-8 -*-
from osv import osv
from osv import fields

class module_estudiante_tipo(osv.osv):
    _name = "module_estudiante.type"
    _inherit = "module_persona.type"
    _description = "Estudiante"
    _columns = {
        'note': fields.char('Nota', size=64, required=True,help='Nota del Estudiante en una Asignatura'),
        'asig': fields.char('Asignatura', size=64, required=True,help='Asignatura en la que se evaluo'),
        'cargo':fields.many2one('module_cargo_estudiante.type','Cargo',select = True,help='Cargo del Estudiante')

    }
module_estudiante_tipo()
```

Figura 26 Integración entre componentes.

Realizada la llamada se obtiene el siguiente resultado:

Figura 27 Resultado de la integración entre componentes.

3.6 Conclusiones parciales

Una vez realizadas las pruebas, se considera que las decisiones arquitectónicas tomadas con respecto a la plataforma de desarrollo fueron satisfactorias, como constancia de esto, son los resultados de las pruebas de concepto, a manera de validación teórica.

Además para lograr una mayor validez se realizaron pruebas de interoperabilidad, seguridad, extensibilidad, y escalabilidad a OpenERP a partir de escenarios descritos de forma práctica.

Los resultados arrojados fueron evaluados de buenos, dado que el sistema permitió agregar nuevos módulos, no permitió la entrada a usuarios no autorizados, se logró la interoperabilidad entre OpenERP y el Sistema de Gestión de Mantenimiento Vehicular que valida que la arquitectura de Cedrux permite la integración con OpenERP.

Conclusiones Generales

- ✓ Se realizó un estudio detallado del estado del arte de la disciplina arquitectura del software, de diferentes sistemas y tecnologías para lograr la definición de la base tecnológica que permite el desarrollo del CRM.
- ✓ Se desarrolló una descripción de las vistas arquitectónicas basándose en el modelo arquitectónico propuesto por el CEIGE.
- ✓ Se seleccionó el método SAAM para la evaluación de la arquitectura basada en escenarios, aplicándose de manera satisfactoria y mostrando buenos resultados.
- ✓ Se realizaron pruebas de interoperabilidad, escalabilidad, extensibilidad y seguridad a la solución a partir de los escenarios descritos, obteniéndose buenos resultados.
- ✓ Se obtuvo una línea base de la arquitectura válida, apoyándose en el modelo arquitectónico propuesto por CEIGE.

Recomendaciones

- ✓ Actualizar el artefacto matriz de integración, en el transcurso del desarrollo del sistema CRM.
- ✓ Realizar otros tipos de pruebas de seguridad a la aplicación.
- ✓ Probar la interoperabilidad del consumo de servicios del OpenERP por parte del sistema Cedrux, como medida proactiva ante la posible integración en este sentido.

Bibliografía

2010 Análisis de los sistemas de Business Intelligence y su aplicación practica en los proyectos de softwareMadrid 2010

Andina, Mind. 2009. Guía Práctica para el Uso del Servicio de. Colombia : s.n., 2009.

2004 Arquitecturas de Software. Guía de Estudio 2004

Bacula. 2012. Bacula. [En línea] 22 de 3 de 2012. http://www.bacula.org/es/.

Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004. Arquitecturas de Software. Guía de Estudio. 2004.

Carlos Reynoso, Nicolás Kiccillof. 2004. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Buenos Aires : s.n., 2004.

2010 Comparativa B.I. Open Source.Barcelona 2010

2011 CRM, el desafío de los clientesBarcelona 2011

Fredricks, Karen S. 2009. SugarCRM For Dummies. Canada : s.n., 2009. 978-0-470-38462-6.

2010 Guía ODBC y JDBC 2010

Hernández, Cisneros. 2008. Interoperabilidad de Módulos de Sistemas R/3 de SAP. 2008.

2012 IBM - Patterns: Service Oriented Architecture and Web Servicehttp://publib-b.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246303.html?Open. SG24-6303-00

2009 Informe de evaluación de ERP 2009

2010 Introducción a BPMN 2010

2004 Introducción a la Arquitectura de SoftwareBuenos Aires 2004

Jeissy Alexandra Londoño Guzmán, Germán Sebastián Cadavid Loaiza, Steven Taborda Ospina, Eliana Giraldo Salazar, Lucas Palacio, Jhonny Ríos Ríos, David Alberto Loaiza Rojas. 2009. MANUAL DE OPENFIRE. ADMINISTRACION DE REDES DE COMPUTADORES. [En línea] 2009. [Citado el: 4 de 12 de 2011.] http://es.scribd.com/doc/8679119/Manual-de-Openfiredoc.

Kazman, Rick, y otros. 2009. SAAM: A Method for Analyzing the Properties of Software Architectures. Center for Systems and Software Engineering. [En línea] 2009. [Citado el: 06 de 04 de 2012.]

Martin, Robert C. Design Principles and Design Patterns.

MDO Tecnologíahttp://blogs.on-site.com.mx/archives/309

Microsoft. 2008. Microsoft Dynamics CRM. [En línea] 2008. [Citado el: 24 de 11 de

2011.] http://www.microsoft.com/latam/dynamics/crm/datasheet/default.aspx.

2011 Modelo de referencia para el desarrollo arquitectónico de sistemas de software en dominios de gestiónLa Habana 2011

OpenIMhttp://www.openim.techlab.smk.fres

Postbank, German. 2012. vtiger. [En línea] 2012. [Citado el: 22 de 1 de 2012.] http://www.vtiger.com/crm.

2010 Prototipo de herramienta de calidad de datosMadrid 2010

Salesforce. 2008. http://www.salesforce.com/crm/. [En línea] 2008. [Citado el: 6 de 12 de 2011.] http://www.salesforce.com/crm/.

2008 Simple object Access Protocol (SOAP) 2008

Sourceforge. 2012. XMPPPY. [En línea] 20 de 1 de 2012. http://xmpppy.sourceforge.net/.

Tahirí Rivero Alvarez, Osmar Leyet Fernández. 2010. GESTIÓN DEL CONOCIMIENTO EN EL EQUIPO DE ANÁLISIS DEL PROYECTO ERP-CUBA. La Habana : s.n., 2010.

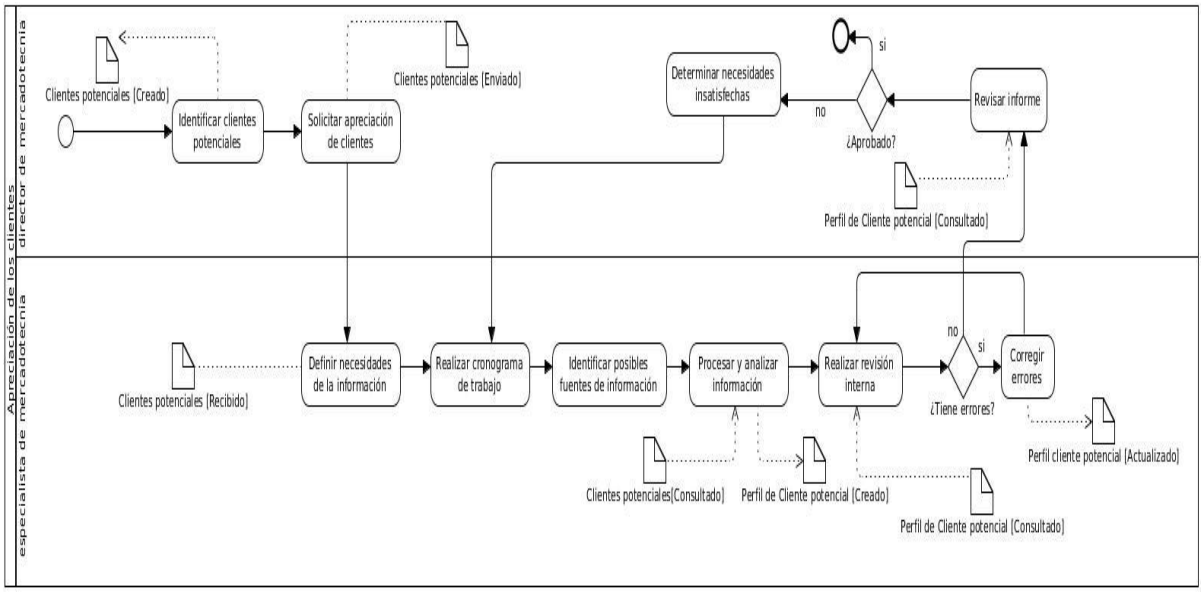
Vásquez, Juan Miguel. 2009. Gestion de la realación con el cliente. Bogota : s.n., 2009.

Villacampa, José Luis. 2008. Estudio integración sistema de mensajería instantánea en plataforma comunicaciones unificadas UPCcom. Barcelona : s.n., 2008.

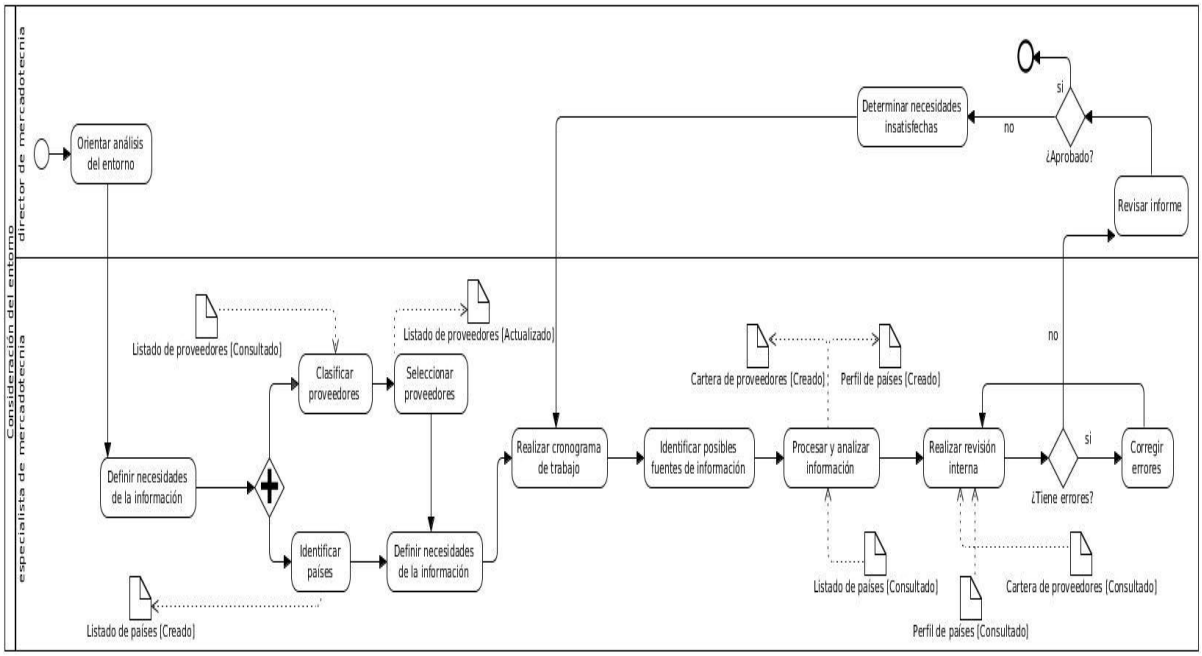
W3C. 2011. HTTP - Hypertext Transfer Protocol. [En línea] 2011. [Citado el: 10 de 03 de 2012.] http://www.w3.org/Protocols/.

Anexos

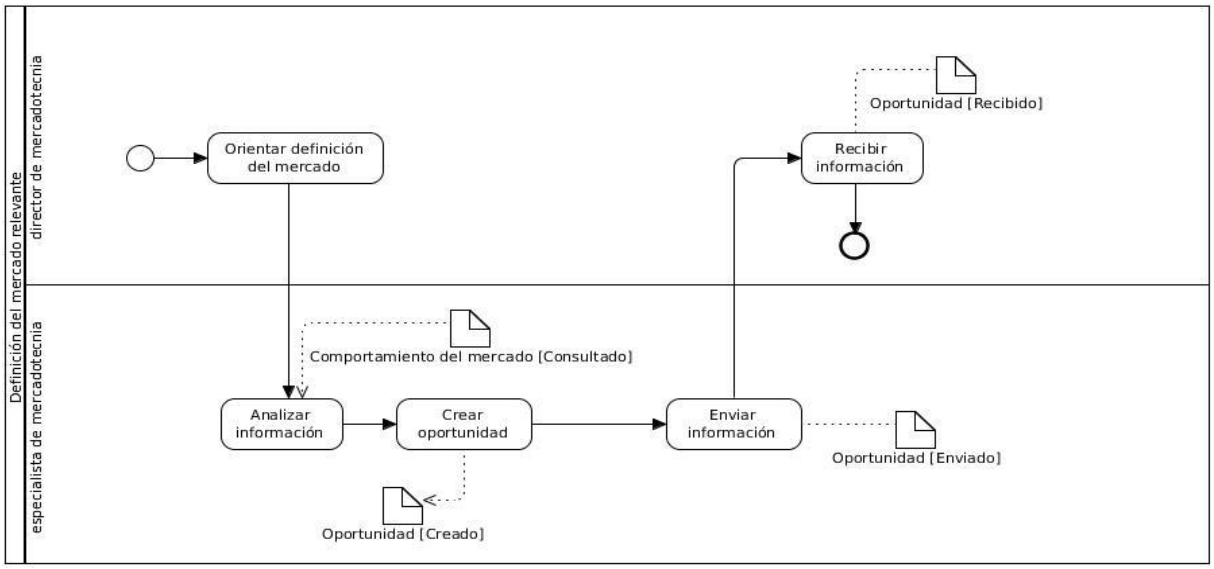
Anexo 1 Análisis del mercado.



Anexo 2 Descripción del subproceso apreciación de los clientes.

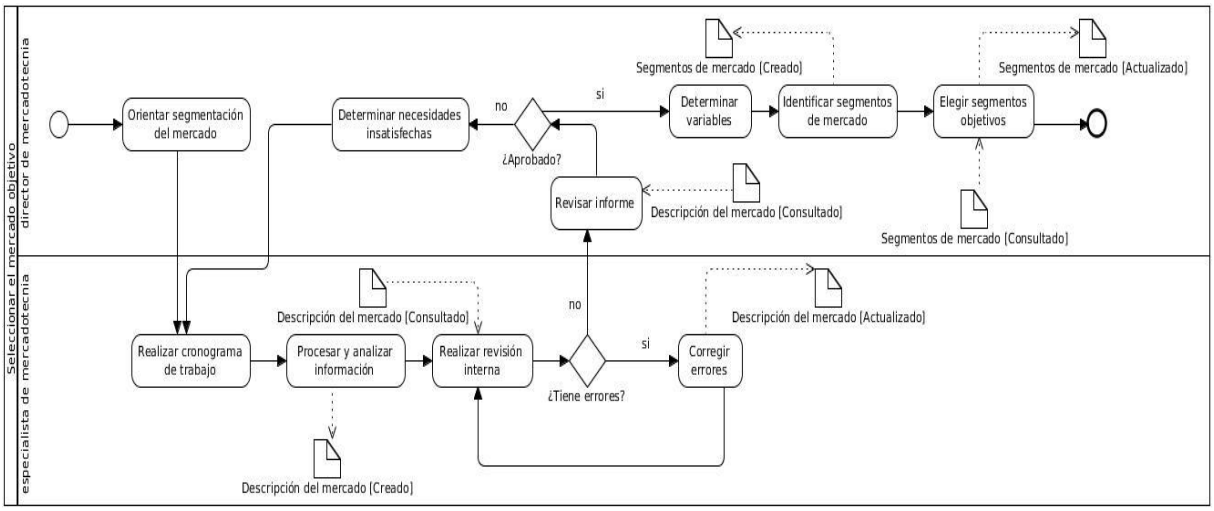


Anexo 3 Descripción del subproceso consideraciones del entorno.



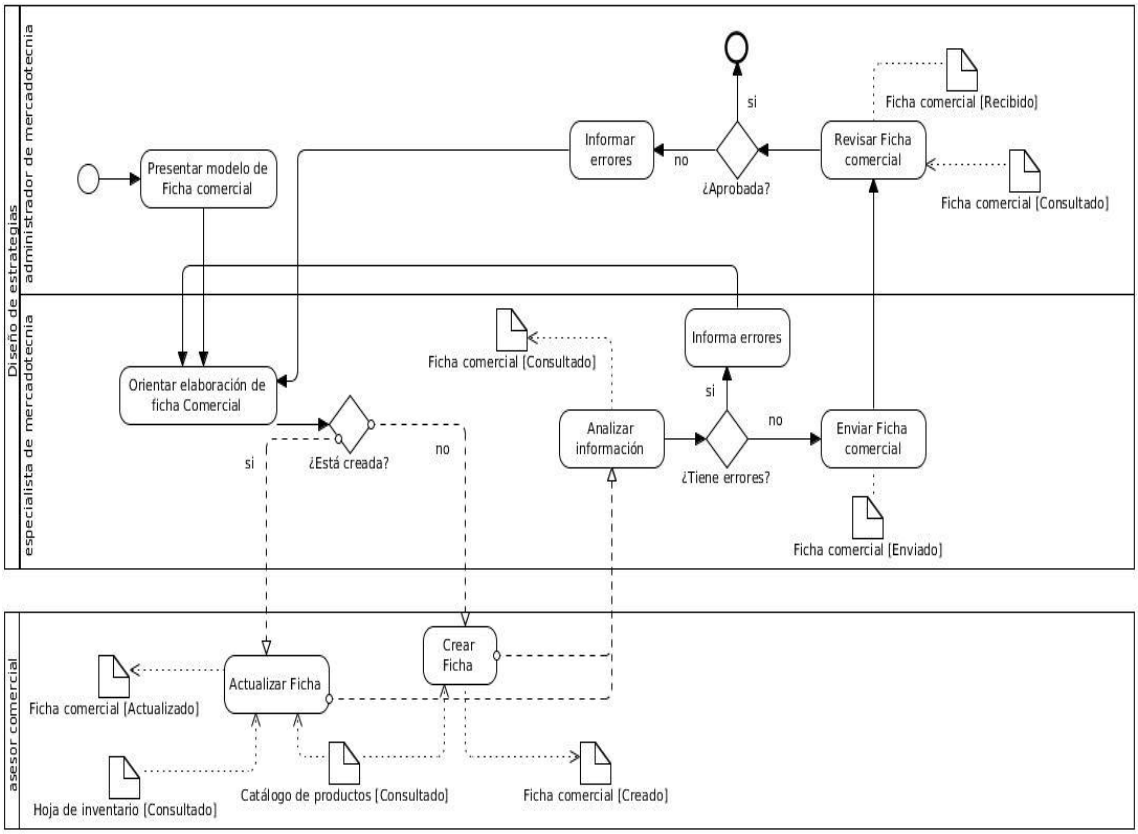
Anexo 4 Descripción del subproceso definición del mercado.

Anexo 5 Selección del mercado objetivo.



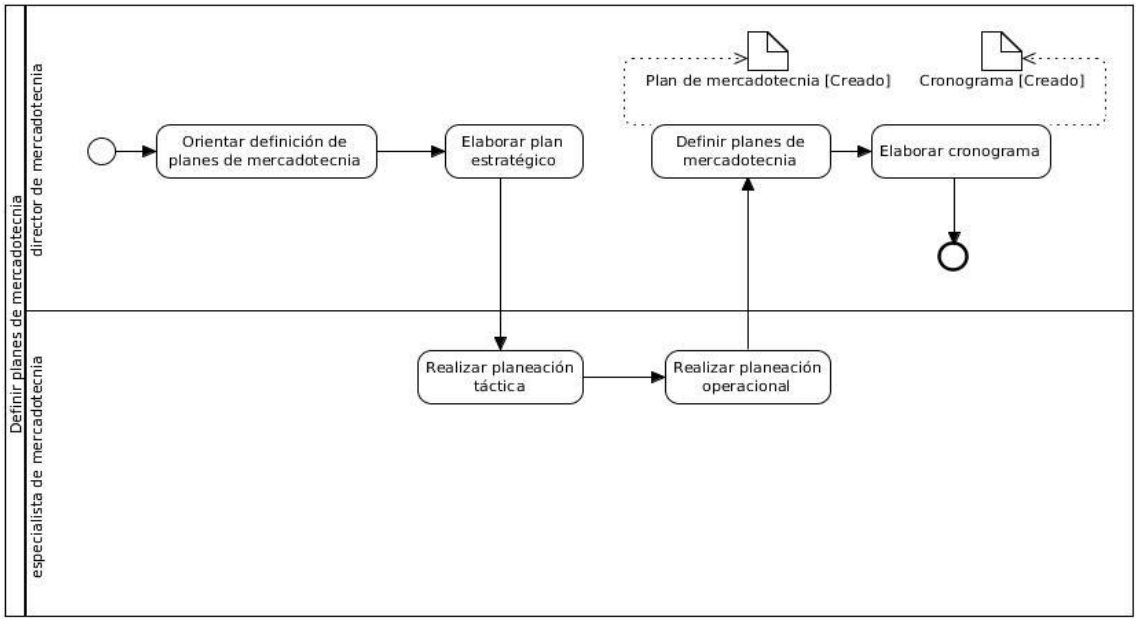
Anexo 6 Descripción del subproceso selección del mercado objetivo.

Anexo 7 Diseño de estrategia del mercadotecnia.



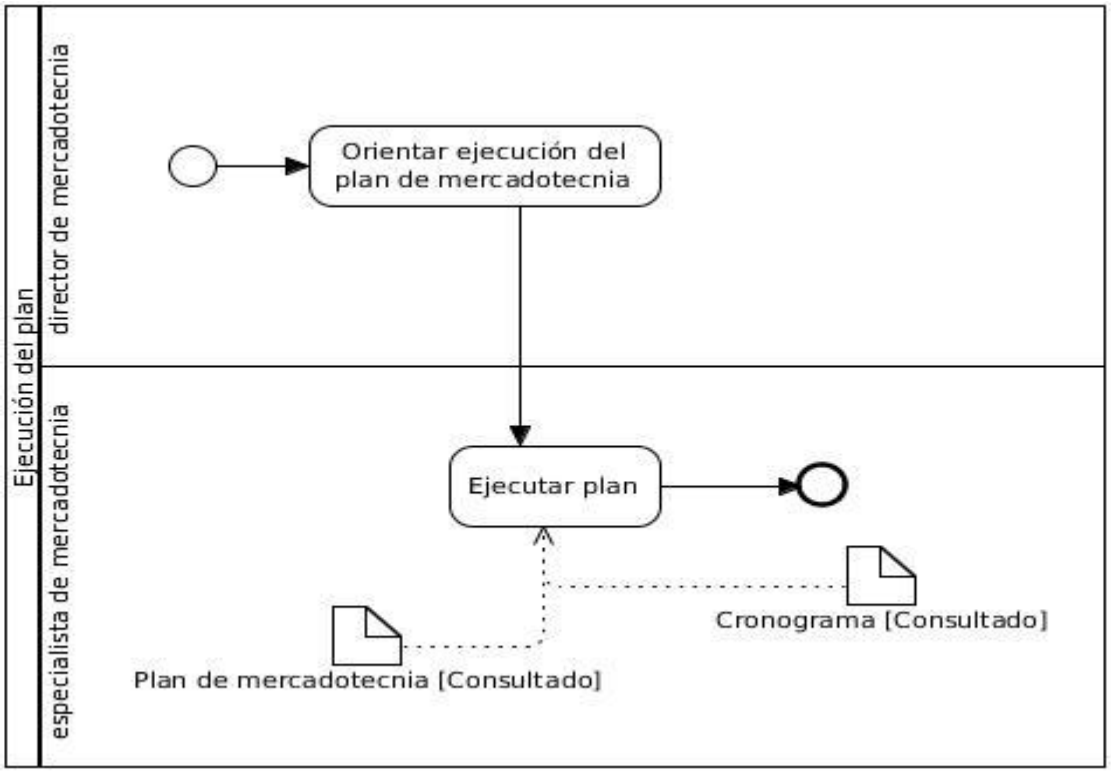
Anexo 8 Descripción del subproceso diseño de estrategia de mercadotecnia.

Anexo 9 Definir planes de mercadotecnia.



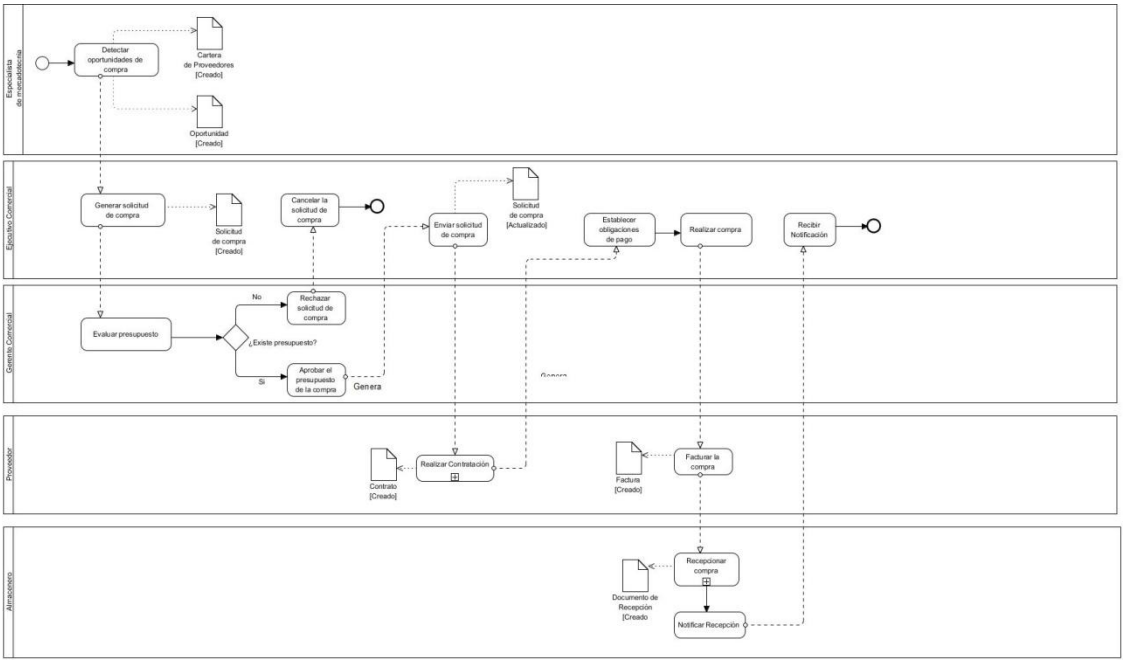
Anexo 10 descripción del subproceso definir planes de mercadotecnia.

Anexo 11 Ejecución del plan.

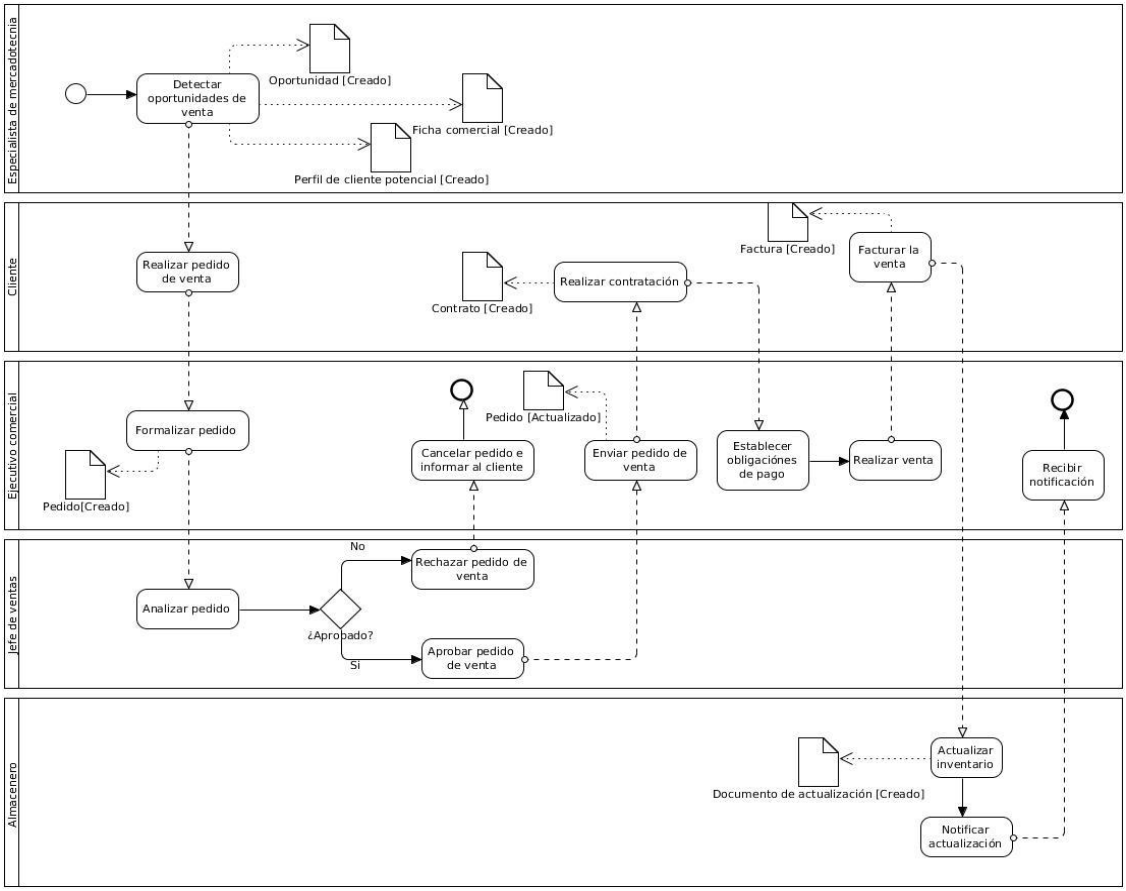


Anexo 12 Descripción del subproceso ejecución del plan de mercadotecnia.

Anexo 13 Compra/Venta.

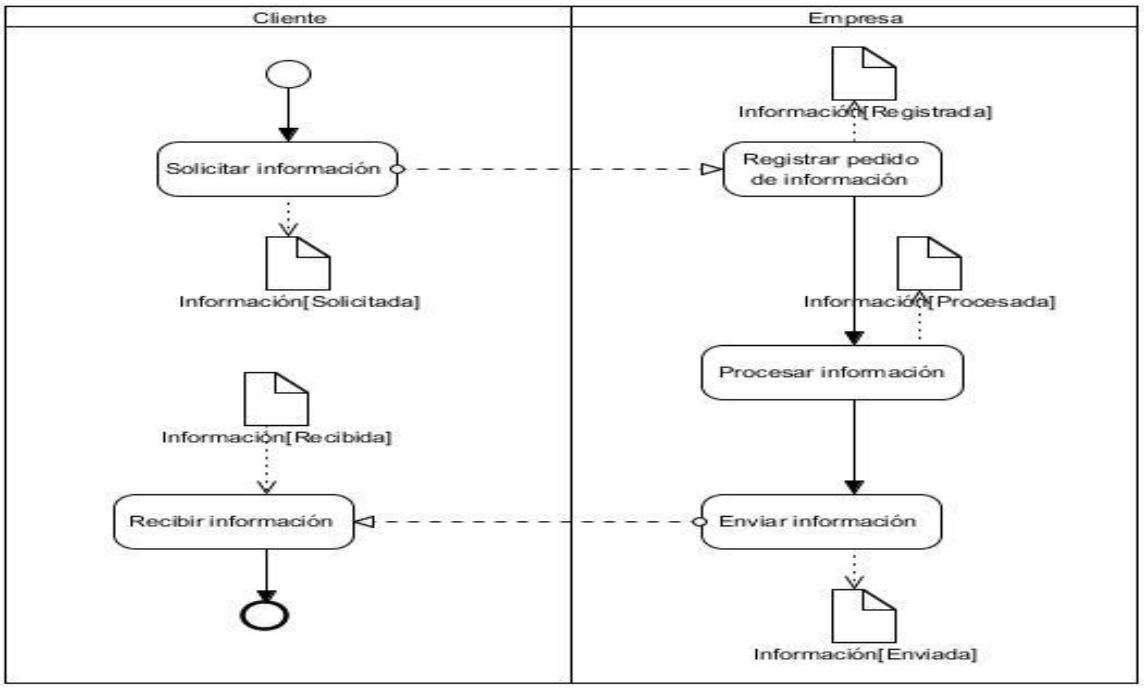


Anexo 14 Descripción del subproceso compras.

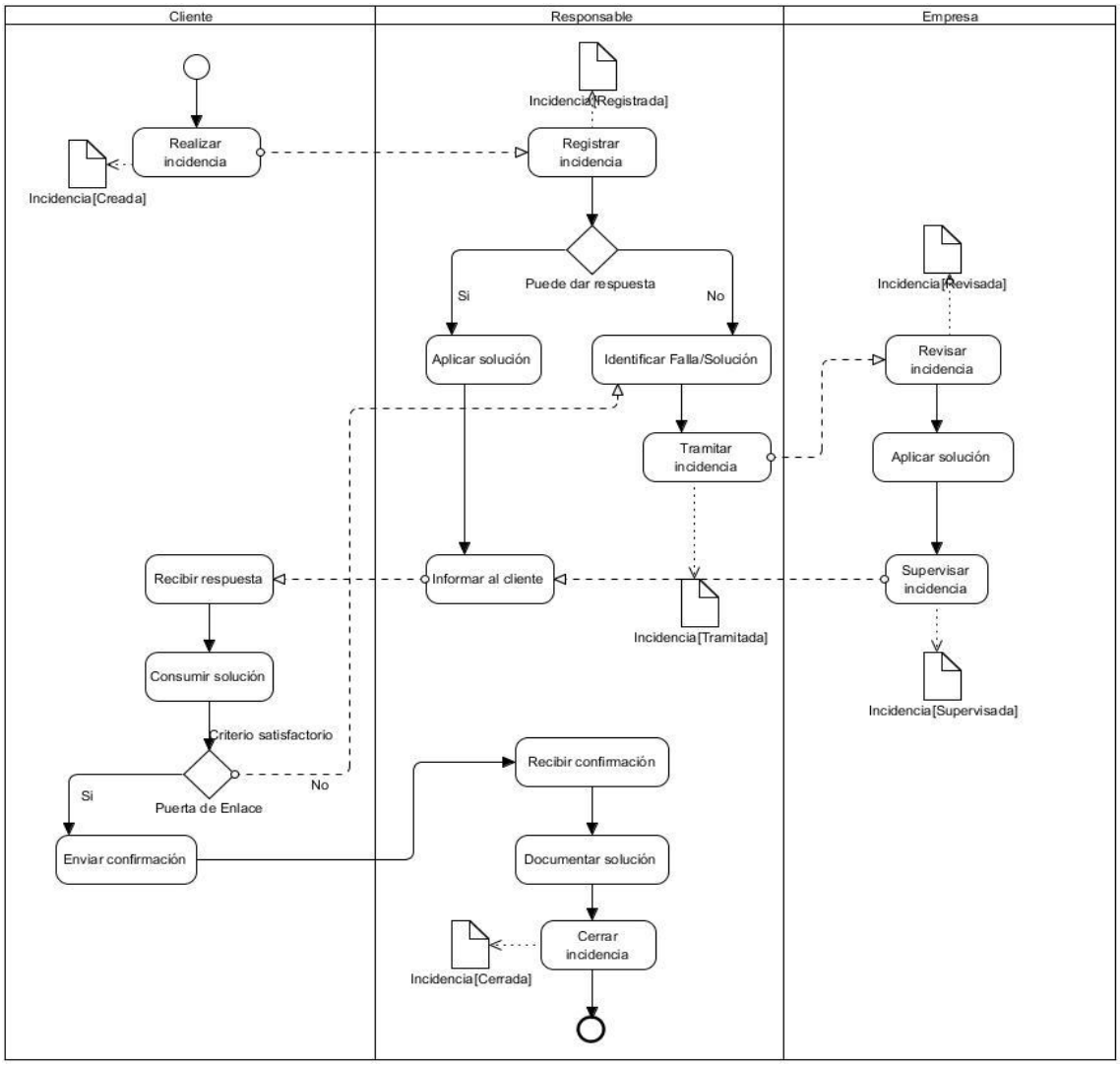


Anexo 15 Descripción del subproceso ventas.

Anexo 16 Atención al cliente.

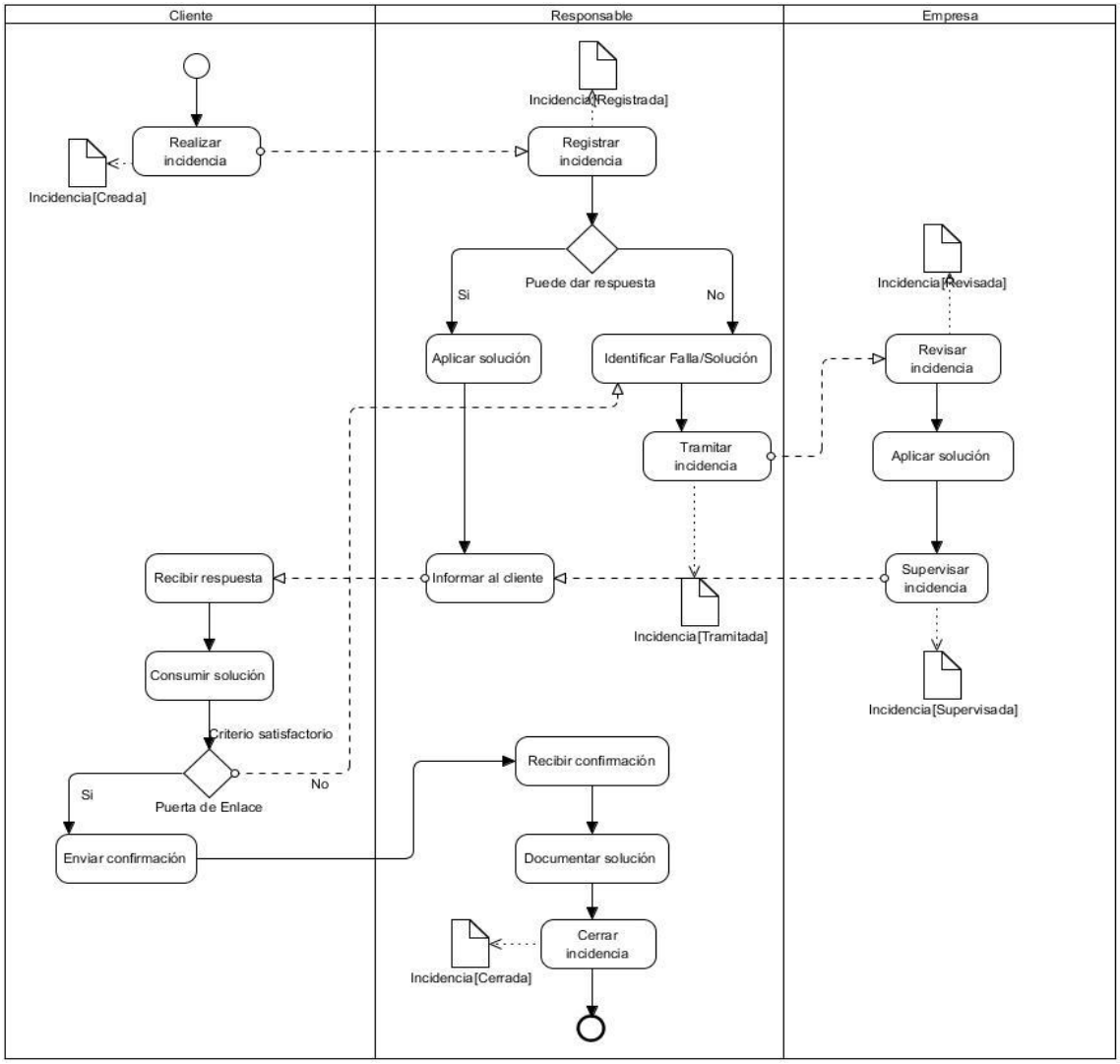


Anexo 17 Descripción del subproceso Información de bienes y servicios.



Anexo 18 Descripción del subproceso gestión de incidencias.

Anexo 19 Medición de la calidad percibida.



Anexo 20 Descripción del subproceso gestión de incidencias.