PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMA DE CONTROL, GESTIÓN Y ADMINISTRACIÓN DEL SERVICIO DE TAXI

TESIS PARA OPTAR EL TÍTULO DE

INGENIERO INFORMÁTICO.

PRESENTADO POR

Hernán Alejandro Quintana Cruz

LIMA – PERÚ

2011

RESUMEN

Este tema de tesis tiene como objetivo presentar los lineamientos para la implementación de una solución que, utilizando teléfonos móviles como medio de comunicación entre participantes, soporte el servicio que ofrece una empresa de taxi.

En el capítulo 1 se da una breve introducción sobre el trabajo realizado así como también se definen los objetivos del proyecto y el alcance del mismo.

En el capítulo 2 se analiza la situación del servicio de transporte de taxis en el Perú, así como los procesos más importantes que las empresas llevan a cabo. Se concluye describiendo el problema a solucionar con la implementación de la aplicación mencionada en esta tesis.

En el capitulo 3 se hace una breve explicación sobre las diferentes tecnologías que se utilizan en la actualidad para el desarrollo de aplicaciones con dispositivos móviles. Se realiza una comparación entre las plataformas de desarrollo de aplicaciones móviles más importantes del mercado: J2ME y BREW.

En el capitulo 4 se procede con el análisis de la aplicación a desarrollar especificando requerimientos y casos de uso que se tomaron en cuenta para el desarrollo de la solución.

El capitulo 5 explica el proceso de diseño poniendo énfasis en la arquitectura en la cual se enmarca la solución.

El capitulo 6 contiene los detalles de la implementación de la aplicación así como los casos de prueba desarrollados para asegurar la calidad y el buen funcionamiento de la solución.

Para finalizar, en el capitulo 7 se exponen las conclusiones a las que se llegó luego del desarrollo del proyecto así como recomendaciones y sugerencias que deberían tomarse en cuenta en desarrollos y/o posteriores ampliaciones.

DEDICATORIA

Mis abuelitos: Alejandro y Leopoldo

Mis abuelitas: Irene y Epifanía

Mis padres: Camilo y Lourdes

Mi hermano Diego

AGRADECIMIENTOS

Es difícil poder nombrar a todas las personas que contribuyeron de una u otra manera en el desarrollo de esta tesis, así que si me olvido de alguno, espero puedan disculparme.

Primero gracias a mi familia, mis abuelos, mis tíos, mis primos, que siempre estuvieron ahí para enseñarme, apoyarme y exigirme cuando era necesario. En especial, este agradecimiento va para mi abuelito Alejandro que me apoyó y aconsejó en todo momento. Muchas gracias por todo Papito, me harás mucha falta.

A mis padres Camilo y Lourdes que estuvieron siempre ahí, apoyándome y enseñándome los fundamentos de la vida.

A Diego que siempre colaboró y me apoyó en todo, ¡eres un maestro hermano!

Muchas gracias también a Rosario, sin sus consejos, ayuda y apoyo esto no se hubiera podido realizar.

A mi asesor Jorge Berrocal y en general, a todos los profesores de la PUCP que contribuyeron con mi formación profesional.

También toca agradecer a mis socios y amigos de toda la vida Iván Herrera y Humberto Flores. Muchas gracias por su colaboración, este proyecto también es de ustedes.

A mis amigos de infancia, de colegio, de universidad, compañeros de trabajo en UNIQUE, IBM pero en especial los de DEVOS ; que en este andar me ayudaron con sus opiniones e ideas. Todo esto no hace más que reforzar en mí la esperanza de que podemos ser grandes como país.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

1. Introducción

En la actualidad, el servicio de taxis en la ciudad de Lima pasa por una situación crítica debido a la gran cantidad de unidades que laboran de manera informal. Esta informalidad trae consigo muchos problemas ya sea para los clientes de este servicio, como también para los propios taxistas. Estos problemas vienen desde los robos, agresiones, secuestros hasta incluso los asesinatos.

Una alternativa para mitigar este problema es la progresiva formalización del servicio de taxis procurando la creación de empresas que garanticen la seguridad y el buen servicio a los clientes. Para ello es necesario facilitar la comunicación entre los clientes y las empresas de taxi.

Una alternativa interesante a considerar para lograr facilitar la comunicación entre clientes y empresas de taxi es el uso del telefonía móvil. Actualmente, el mercado de la telefonía móvil viene creciendo vertiginosamente dado el interés de la población en obtener estos aparatos. Si bien es cierto, en un primer momento el alcance de estas

tecnologías solo se limitaba a determinados niveles socioeconómicos, en la actualidad su uso se ha generalizado permitiendo mayores posibilidades de desarrollo de nuevos negocios relativos a este mercado.

Además, el gran desarrollo y la alta difusión que han alcanzado estos dispositivos nos permiten tener una gran capacidad de procesamiento así como una buena respuesta en la ejecución de aplicaciones de cierta complejidad. Todo esto permite contar con nuevas herramientas que mejorarían la relación cliente-empresa contando con un mercado objetivo (cliente con teléfono móvil) bien definido.

Aplicar las tecnologías móviles a la problemática del servicio de taxis permitiría una mejora en las comunicaciones entre los clientes y la empresa de taxi, así como contribuiría a su organización. Dado estos beneficios, aumentaría la seguridad y por ende, el nivel de consumo en este mercado, favoreciendo a la propia empresa de taxi, así como a toda la ciudadanía.

1.1. Objetivos y alcances

Esta tesis tiene como objetivo presentar los lineamientos para la implementación de una solución que, utilizando teléfonos móviles como medio de comunicación entre participantes, que mejore el servicio que ofrece una empresa de taxi.

El sistema contará con los siguientes módulos:

- **Aplicación Móvil.** La aplicación permitirá el registro de pedidos que serán enviados por los clientes ingresando la información necesaria para que la empresa de taxi elija un taxista y envíe la oferta correspondiente. Posteriormente, el cliente podrá elegir la oferta que más se adapte a sus necesidades. Además la aplicación permitirá la consulta de servicios que se encuentren en ejecución. Adicionalmente, se contará con una sección de historial de servicios que permitirá a los clientes poder consultar los servicios pasados en los que hayan incurrido.

- **Sistema de Gestión Central.** Este módulo de la aplicación será el encargado de la inscripción de las empresas de taxi que participaran en las subastas de pedidos y que recibirán la información relativa al negocio. También este módulo se encargará de la gestión de pedidos, permitiendo: mostrar los pedidos y permitir a las empresas de taxi generar ofertas; mostrar las ofertas que han sido aceptadas por los clientes para procesarlas y posteriormente poder darles de baja luego de la finalización del servicio. Además, se encargará de generar los reportes referentes al servicio dado mostrando información respecto al número de servicios por lugar geográfico, número de servicios por compañía, precios promedio, etc.

2. Marco conceptual

Actualmente el servicio de taxi que se brinda en nuestra ciudad es de dos clases: taxista independiente y empresa proveedora de servicios de taxi, ambos con su respectiva problemática.

En el caso del taxista independiente, el servicio que se brinda no provee al cliente las medidas necesarias para garantizar la seguridad tanto del chofer como del cliente. Este servicio se realiza mediante un convenio informal sin dejar constancia física o documentación alguna del servicio brindado. La calidad del servicio depende únicamente del chofer y en muchos casos los clientes no tienen opción a quejarse o a evitar el uso posterior del mismo servicio brindado por el mismo chofer.

En el caso de las empresas de taxi, el servicio se brinda mediante un convenio sin un contrato previo, entregándose una constancia del servicio solo al terminar dicho servicio (boleta u otros). Esta modalidad tampoco brinda todas las garantías de seguridad requeridas, ya que el cliente no recibe información alguna sobre el chofer que será asignado a su pedido y que no se brinda garantía alguna de que se reciba el servicio solicitado.

Esta clasificación hace que el negocio de taxis sea especial dentro del rubro de transporte, dado que cada servicio influye decisivamente en el nivel de satisfacción que el cliente tiene con la compañía.

Los participantes del negocio del servicio de taxis son cuatro: el cliente, la compañía de taxi, el chofer de taxi independiente y el chofer de una compañía de taxi, cada uno con sus respectivas funcionalidades. La compañía es el ente encargado de la gestión del servicio, la logística relacionada a este (vehículos) y los recursos humanos (choferes, empleados administrativos, etc.). El cliente es quien genera y recibe el servicio. El chofer de taxi es el participante encargado de llevar a cabo el servicio.

2.1. Situación del mercado

2.1.1. Servicio de transporte por taxi

Dentro del negocio del transporte, el servicio de taxi es uno de los servicios de uso más generalizado. Los ciudadanos tienden a utilizar este medio para poder movilizarse de un lugar a otro en tiempo menor a los usualmente brindados por el transporte público masivo, así como también por la seguridad que suelen otorgar.

Los problemas actuales por los que pasa el país han repercutido en el funcionamiento del negocio. La falta de estabilidad laboral y la libre importación de automóviles usados ha generado que una gran cantidad de personas opten por el trabajo de taxista, dada la relativa facilidad (en oferta y costo) de conseguir las herramientas para realizar la labor.

Esta excesiva oferta así como los altos costos de operación (licencias de funcionamiento, mantenimiento de los vehículos) han dado paso a una creciente informalidad que repercute negativamente en la calidad del servicio,

ya sea por la falta de seguridad, maltrato o por el excesivo tiempo de transporte.

Una solución natural a esta informalidad es la creación de empresas que se dediquen a brindar el servicio. En el Perú, existen empresas que brindan este servicio, pero debido a que los costos involucrados en que estas incurren son altos, solamente llegan a cubrir un pequeño sector de los usuarios de taxi de l ciudad, dejando la mayoría restante a los taxistas informales.

Realizando un análisis de esta situación, se llega a la conclusión de que el problema de la formalidad en el servicio de taxi es que no existe un mercado establecido, con proveedores de servicio claramente definidos y clientes deseosos de invertir en este servicio. En este seudo-mercado, los proveedores son entes informales que no cuentan con ningún control comercial por lo que los clientes carecen de opciones serias con la cual invertir su dinero (o las que existen son muy caras y no están a su alcance).

2.1.2. Aplicaciones para dispositivos móviles

Actualmente las empresas proveedoras de tecnologías móviles no se orientan al desarrollo de aplicaciones de uso general como se da en otros países con gran impacto comercial. En los Estados Unidos de América y en Europa, existen aplicaciones que van desde el servicio de mapas (Google Maps [GOO01], servicio de guía de teléfonos, planificación de vuelos y compras de productos [GOO02]).

2.2. Procesos del negocio

2.2.1. Participantes en el negocio

Existen cuatro tipos definidos de participantes en el negocio del servicio de taxis (Figura 2.1).

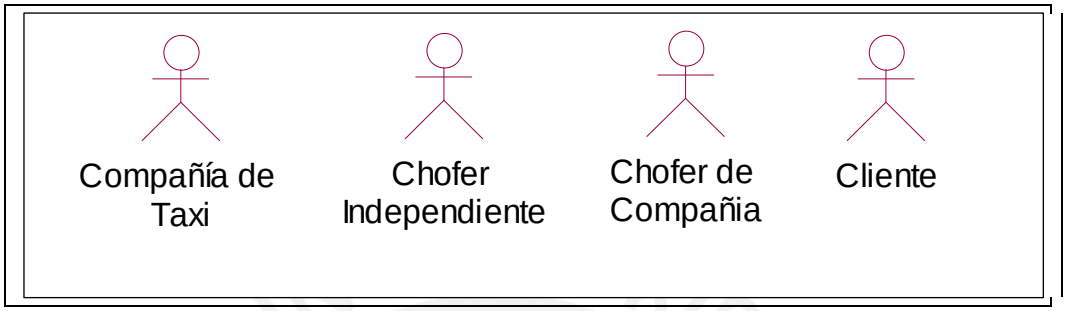

FIGURA 2.1 Participantes del negocio

- **Cliente.** Persona natural o jurídica que realiza el pedido de un taxi para trasladarse al lugar que desee. Las necesidades que tiene se enfocan a una correcta realización del servicio por parte de la compañía de taxi considerando factores como seguridad, puntualidad y la tarifa del servicio.

- **Chofer Independiente.** Persona natural encargada de la realización del servicio. Generalmente es un individuo particular sin relación laboral con alguna empresa de taxi.

- **Chofer de Compañía.** Persona natural encargada de la realización del servicio. Este pertenece a una compañía que gestiona los pedidos realizados por el Cliente.

- **Compañía de Taxi.** Persona jurídica encargada de la gestión del servicio de taxi. Cuenta con un grupo de unidades (vehículos) a su cargo a las cuales se les asigna un chofer. Es la unidad que se encarga de la correcta administración y funcionamiento de los procesos relacionados al servicio de taxi.

2.2.2. Generar pedido

Descripción:

Este proceso se inicia con la necesidad del Cliente de pedir un servicio de taxi para poder trasladarse de un lugar a otro dentro de la ciudad.

Lo primero que el cliente debe decidir es si desea hacer el pedido a un Chofer independiente o a una Compañía de taxis.

Para el caso que el Cliente decida utilizar los servicios de un Chofer independiente (figura 2.2), el pedido se realiza de forma personal directamente entre el Cliente y el Chofer.

El Cliente detiene un vehículo de taxi independiente y entrega los datos iniciales del pedido al chofer, como el lugar destino y una posible ruta especifica que desee. El Chofer independiente evalúa el pedido y hace una oferta de tarifa. El Cliente decide si acepta o no dicha oferta.

Si el Cliente decide consultar a una Compañía de taxi (figura 2.3), la única manera de pedir este servicio es por medio de la línea telefónica.

El Cliente escoge a una Compañía de taxis y efectúa su pedido indicando el día, hora, lugar de recojo y el destino, luego la Compañía de taxi responde al pedido con una cotización basándose en la información proporcionada por el Cliente. Si la cotización que recibe el Cliente es satisfactoria, entonces este la acepta y la Compañía le entrega información adicional del servicio que podría ser el tipo de vehículo, el nombre del chofer, etc. El Cliente y la Compañía de taxis llegan a un acuerdo sobre la hora y el lugar desde donde se comenzará a realizar el servicio.

El proceso termina cuando el cliente acepta alguna oferta o si decide no utilizar servicios de taxi.

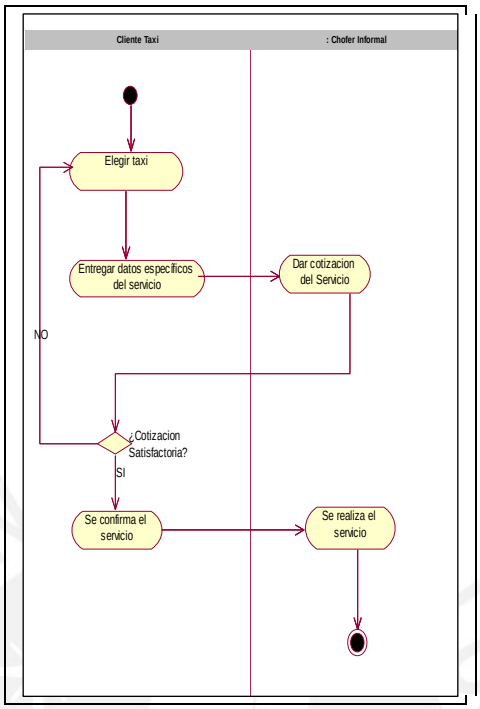

FIGURA 2.2 Generar pedido (chofer independiente)

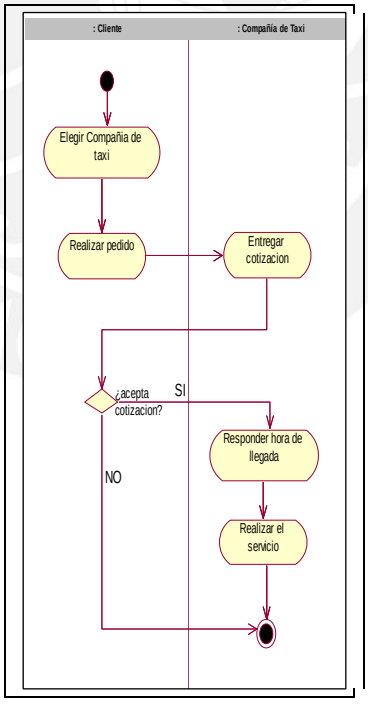

FIGURA 2.3 Generar pedido (chofer de compañía de taxi)

2.2.3. Notificar servicio a taxista

Descripción:

Para la realización de este proceso, es necesario que se haya efectuado anteriormente el proceso Generar Pedido, optando por la opción Elegir Compañía de taxis.

A continuación se presenta el diagrama de actividades relativo a este proceso (Figura 2.4).

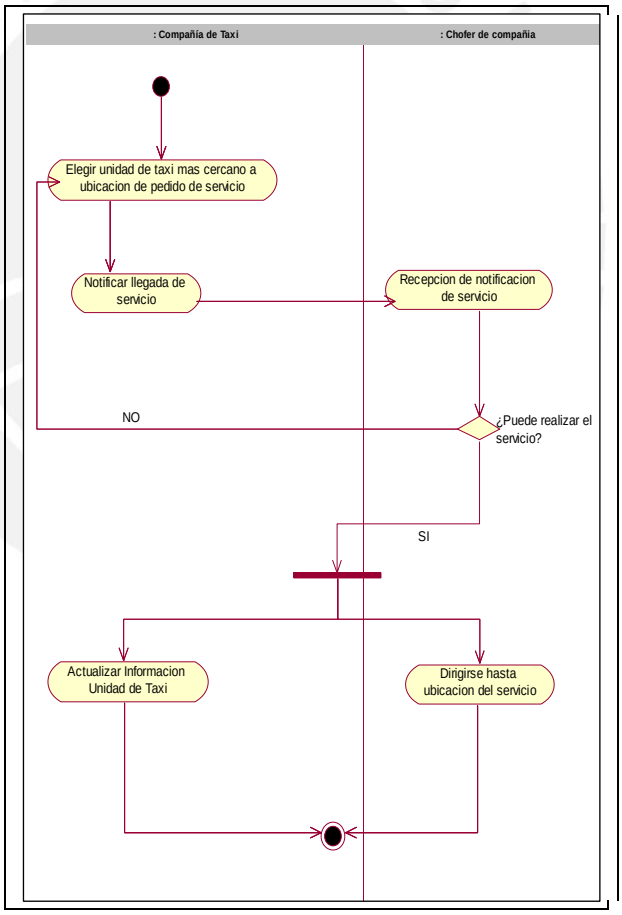

FIGURA 2.4 Notificar servicio a taxista

El proceso se inicia cuando la Compañía de taxi tiene un pedido pendiente de atención y asignación de Chofer. Se procede a ubicar las unidades de taxi más

cercanas al punto de recojo del cliente, por lo que es muy importante que la Compañía cuente con el estado actualizando de sus unidades (ver Sección 2.2.5 Actualizar estado de unidad de Taxi por parte de Compañía).

Una vez que la Compañía ha seleccionado la unidad más cercana, se procede a notificar al Chofer sobre la llegada del pedido y las características relativas a éste.

Si el Chofer puede realizar el servicio (basándose en factores como suficiente cantidad de combustible, buen estado de la unidad, buen estado anímico del mismo), la compañía de taxi pasa a actualizar la información de la unidad de taxi.

En el caso que el chofer acepte realizar el servicio, debe dirigirse al punto de recojo del Cliente.

El proceso termina con los datos de la unidad de taxi actualizados en la base de datos de la Compañía.

2.2.4. Finalizar servicio

Descripción:

Para que este proceso se efectúe, se requerirá que el Chofer de la unidad de taxi haya aceptado realizar algún servicio, que se haya llevado al Cliente al punto de destino estipulado y se haya realizado la transacción económica.

El proceso se inicia cuando el Chofer comunica la finalización del servicio a la Compañía de taxi. La Compañía de taxi recibe la notificación de fin de servicio y actualiza la información de la unidad de taxi obteniendo la tarifa del servicio y el tiempo empleado.

Esta información servirá posteriormente para calcular costos y estimar las ganancias obtenidas por los servicios.

A continuación se presenta el diagrama de actividades relativo a este proceso (Figura 2.5).

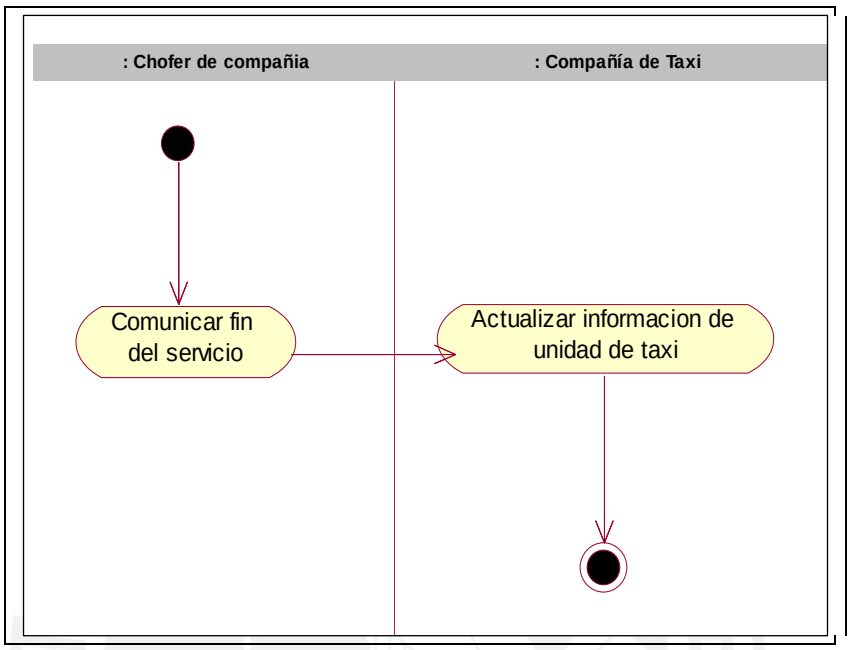

FIGURA 2.5 Finalizar servicio

2.2.5. Actualizar estado de la unidad de taxi por parte de la Compañía de taxi

Descripción:

El objetivo de este proceso es que la Compañía de taxis pueda actualizar la información y el estado de las distintas unidades de taxi con las que cuenta, para que posteriormente se pueda evaluar y elegir de entre las unidades disponibles cuando se requiera brindar un servicio.

El diagrama de actividades de este proceso se presenta en la Figura 2.6

El proceso se inicia cuando la Compañía de taxi elige una unidad de la lista para poder actualizar su información.

La unidad le responde si se encuentra brindando algún servicio o no. En el caso que se encuentre brindando servicio, ésta indicará su información geográfica e informará acerca del servicio que se encuentra realizando. Si la unidad no se encontrara brindando servicio alguno, entonces solamente deberá indicar su ubicación geográfica.

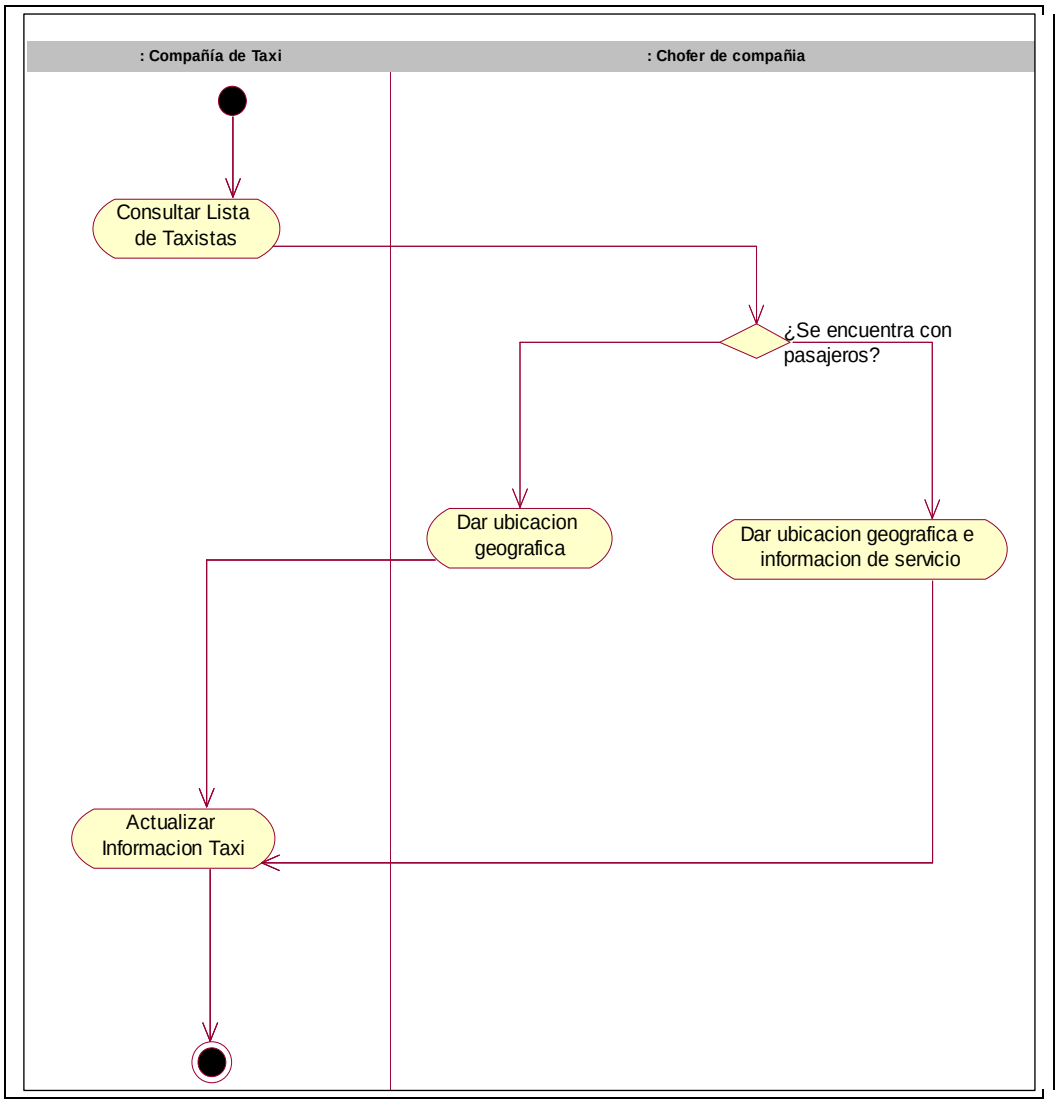

FIGURA 2.6 Actualizar estado de la unidad de taxi por parte de la Compañía de taxi

Cabe resaltar que las unidades de taxi no solo atienden servicios derivados exclusivamente de la central de la compañía, sino que también se les permite recoger pasajeros en la calle sin previa notificación de pedido.

2.2.6. Actualizar estado de unidad de taxi por parte del chofer

Descripción:

El objetivo de este proceso es que el chofer pueda notificar su estado actual a la Compañía.

Este proceso, al igual que el llamado “Actualizar estado de unidad de taxi por parte de la Compañía”, es necesario para mantener actualizada la información que posee la Compañía de sus unidades dado que, como anteriormente se mencionó, las unidades pueden recoger pasajeros sin necesidad de que hayan hecho un pedido a la compañía.

El proceso se inicia cuando el Chofer comunica su estado a la Compañía de taxi como se aprecia en la Figura 2.7. Entonces la Compañía actualiza el estado de dicho Chofer en su base de datos.

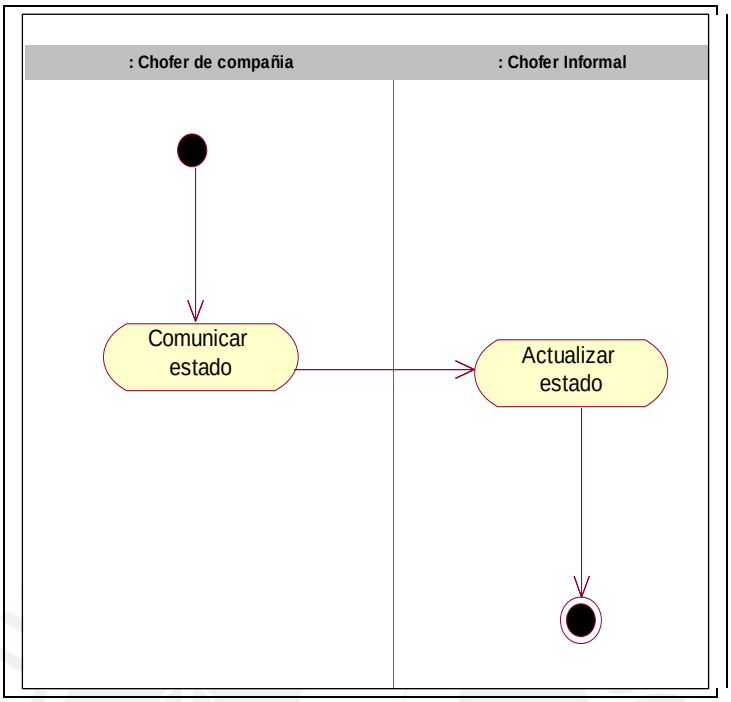

FIGURA 2.7 Actualizar estado de unidad de taxi por parte del chofer

2.2.7. Cancelar pedido

Descripción:

Este proceso permite la cancelación de un pedido por parte del cliente. Cabe resaltar que por motivos de costos se determina un tiempo máximo el cual podría transcurrir entre la generación del pedido y la cancelación de éste.

El proceso comienza con la intención del Cliente de cancelar su pedido como se aprecia en la Figura 2.8. El cliente entrega sus datos personales y los datos del pedido (lugar de origen, lugar de destino, fecha y hora de recojo y otros que se consideren importantes) para que la compañía pueda buscarlo.

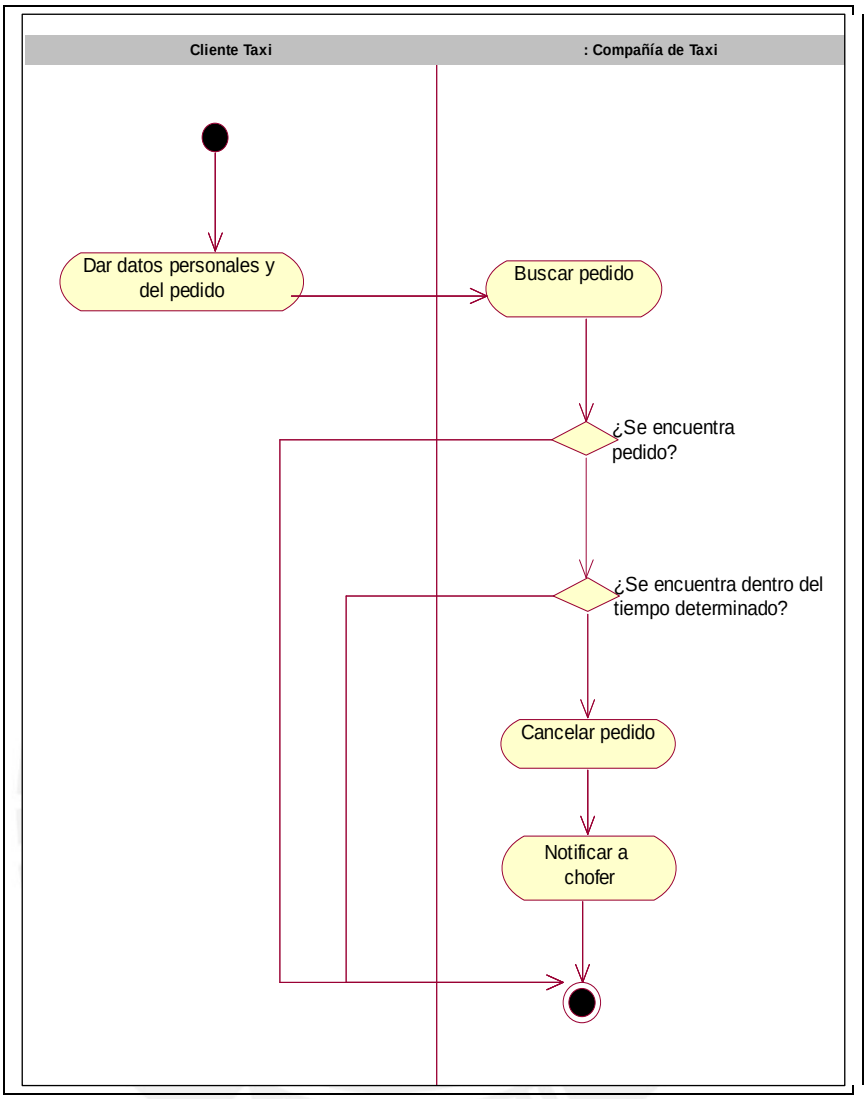

FIGURA 2.8 Cancelar pedido

En el caso que se encuentre el pedido, se consulta la hora para poder determinar si se procede a la cancelación del pedido. En el caso afirmativo se cancela el pedido.

2.3. Problemática del negocio

El principal problema encontrado dentro del servicio de transporte de taxis es la gran brecha existente entre las compañías de taxi y los clientes. Para un cliente

que busca un servicio de taxi de una empresa, éste suele aceptar la oferta dada por la primera con la que se haya comunicado, no permitiendo la libre competencia entre las demás empresas.

Adicionalmente, la falta de definición en los procesos de las empresas de taxi conlleva a la pérdida de información completa y precisa de su negocio, que podría ser utilizada en la mejora de la empresa y en la búsqueda de oportunidades de crecimiento y posicionamiento en el mercado de transporte público. Esto también permitiría mejorar los procesos de la calidad de servicio, al conocerse las necesidades reales del cliente. Un ejemplo claro es el tiempo de llegada de una unidad de taxi al punto de partida del servicio (luego de haberse pactado la realización de este). Este tiempo usualmente toma entre media hora a una hora haciendo esperar a los clientes. Contando con la información de las unidades (ubicación), se podrá disminuir progresivamente este tiempo repercutiendo directamente en la satisfacción del cliente.

Otro problema que se encuentra en el servicio de taxi es la informalidad. Los problemas económicos que aquejan a la mayoría de ciudadanos de nuestro país influyen directamente en el mercado de transporte público de la ciudad. La inseguridad ciudadana, los altos precios en el servicio y la poca difusión de éste repercuten en el ciudadano común y corriente, llevándolo a optar por opciones de servicio de taxi informal que frecuentemente no cubren los estándares de calidad y seguridad de servicio que se requieren, pero que son accesibles a su capacidad de gasto.

En general, el mercado de transporte público de la ciudad de Lima, se encuentra en una situación de informalidad, lo que tiene como consecuencia la pérdida de grandes oportunidades de crecimiento y de mejoras en el servicio.

3. Tecnologías para dispositivos móviles

3.1. Panorama del mercado

En un inicio, los dispositivos de comunicación móviles (teléfonos celulares) sólo ofrecían la facilidad de comunicarse en forma inalámbrica como ventaja sobre el uso de los teléfonos normales. Con el avance de la tecnología, éstos se han convertido en complicados artefactos de cómputo.

Los nuevos dispositivos vienen con la posibilidad de ser programables, dándole a los desarrolladores a nivel mundial, la posibilidad de adicionarles nuevas funcionalidades a éstos.

La evolución de los dispositivos móviles se divide en generaciones: Los dispositivos de Primera Generación (1G), los de segunda generación (2G) y los de tercera generación (3G).

Los dispositivos de la llamada Primera Generación (1G) solamente podían soportar programas que estuvieran embebidos dentro del *firmware* del dispositivo. Además,

estos dispositivos se caracterizaban porque la comunicación era analógica (solamente voz). Debido a la gran demanda de usuarios que generó, la calidad del servicio se fue haciéndose cada vez mas pobre, lo que se solucionó utilizando la tecnología CDMA.

La Segunda Generación (2G) aparece alrededor de 1990 con la entrada del estándar GSM Europeo que utiliza la tecnología TDMA. Una característica resaltante de esta generación es que se comenzó a trabajar con data digital, lo que permitió soportar velocidades altas de información. Otros servicios extras que se ofrecieron fueron la transmisión de datos (aún limitado), fax y SMS (*Short Message Service*).

La Tercera Generación (3G) se caracteriza por unir la comunicación de datos y la de voz, con el acceso inalámbrico a Internet. Nos permite ejecutar aplicaciones multimedia: voz, navegación por la web, e-mail, transferencia de documentos e imágenes fijas, servicios de ubicación, video de alta definición y altas transmisiones de datos.

Nombre	Características
1G 1979	Telefonía celular analógica sólo para voz Poco segura
2G 1990	Telefonía celular analógica Technology GSM (Global System per Mobile Communications) Convergencia de voz y datos Servicios auxiliares: datos, fax y SMS (Short Message Service)
3G	Telefonía celular digital Convergencia de voz y datos, acceso inalámbrico a Internet Protocolo de alta velocidad de información Aplicaciones audio MP3, video en movimiento, videoconferencia en tiempo real y acceso rápido a Internet

CUADRO 3.1 Características importantes por generación

3.2. Análisis de plataformas

Uno de los más grandes problemas en el negocio de las telecomunicaciones, es la falta de estandarización en los dispositivos móviles. Existen diversas formas de clasificar los dispositivos móviles y cada una de estas con sus propios estándares: por generación a la que pertenecen, por características de hardware, por fabricante, etc.

Debido a que el avance en tecnología es continuo y se desarrollan nuevas tecnologías en poco tiempo, la adopción de estándares únicos se ha hecho muy complicada. Además, otro problema que impide la estandarización es que existen en el mercado gran cantidad y variedad de modelos y cada uno de estos con sus respectivas limitaciones de hardware. Sin embargo, las empresas han tratado de definir plataformas dentro de las cuales se podrían desarrollar aplicaciones que fueran portables entre los diferentes modelos de dispositivos.

Por esto, el primer y más grande problema con el que se encuentran los programadores de aplicaciones para dispositivos móviles, es el de decidir con que plataforma se va a trabajar. El mercado de las aplicaciones móviles nos ofrece en la actualidad diferentes opciones, cada una con sus puntos a favor y sus puntos en contra. Es difícil hacer una distinción certera sobre cual es mejor, esto debido a los diversos escenarios que se nos presentan al intentar desarrollar una aplicación móvil.

El propósito de este capítulo es analizar y comparar a profundidad dos de estas plataformas, que son las de mayor uso en la actualidad en el mercado peruano, para así también poder optar por una de ellas

3.2.1. J2ME (Java 2 Micro Edition)

J2ME pertenece a la familia de las tecnologías JAVA. Este conjunto de tecnologías, que utilizan un lenguaje de programación del mismo nombre, fueron desarrolladas por la compañía Sun Microsystems para desarrollar

soluciones informáticas que logren satisfacer tanto necesidades complejas como sencillas de manera eficiente. Hoy en día muchas de las aplicaciones que utilizamos en nuestras computadoras están desarrolladas en base a ésta tecnología y muchas empresas desarrolladoras de software optan por JAVA debido a los beneficios que éstas proveen.

Sin embargo, a pesar de la gran popularidad con que cuentan las plataformas JAVA, algo muy poco conocido es que, en un comienzo, fueron desarrolladas para funcionar con dispositivos electrónicos de capacidades limitadas. Es así como, a principios de los 90s, la empresa Sun Microsystems inicia el desarrolló de la tecnología J2ME con la creación de un nuevo lenguaje de programación al cual llamó *Oak*, el cual fue desarrollado como parte de un proyecto de investigación que buscaba construir artefactos electrónicos que funcionen principalmente por software.

El primer prototipo construido que interpretaba este lenguaje fue un controlador portable llamado *Star7*, un pequeño dispositivo de mano con una pantalla táctil LCD que tenia incorporado soporte a redes inalámbricas y comunicaciones infrarrojas.

Posteriormente, Sun Microsystems renombró al lenguaje de programación Oak como Java.

3.2.1.1. Características:

- Su lenguaje es sintácticamente igual al C++ pero con algunas diferencias importantes como en el uso de punteros. Mientras que en C++ el programador puede utilizar libremente los punteros acarreando serias complicaciones si no se tiene el debido cuidado, la plataforma JAVA se encarga de gestionar las referencias a estos objetos impidiendo un mal manejo de los recursos de memoria.

- El uso de una máquina virtual es uno de los principales fundamentos de esta plataforma. Este tipo de arquitectura ofrece ventajas como la posibilidad de ejecutar aplicativos en entornos diferentes (sea de hardware o sistema operativo) utilizando el mismo código fuente y el mismo código compilado. Esto se da gracias a la posterior interpretación que la máquina virtual dará al código compilado (llamado *bytecode*) que variará según el entorno en que se encuentre.

- Existen gran número de APIs (*Application Programming Interfaces*) que permiten un fácil desarrollo de aplicaciones de distinto propósito.

3.2.1.2. Arquitectura

En cuanto a su arquitectura, la plataforma J2ME esta constituida por 3 capas claramente definidas: La primera capa es llamada JVM (*Java Virtual Machine*), una capa de Configuración y una capa de Perfiles. La *Virtual Machine* se comunica con la capa de Configuración proveyéndole una interfase con el sistema operativo que se encuentra en el dispositivo. Encima de la Virtual Machine se sitúa la siguiente capa, que es la de Configuración, la cual fue diseñada para ser la misma para todos los dispositivos. La última capa es la de Perfiles, que varía según la configuración de hardware.

3.2.1.3. Organización

En cuanto a su organización, la plataforma J2ME se divide en configuraciones, perfiles y paquetes opcionales como se aprecia en la Figura 3.1.

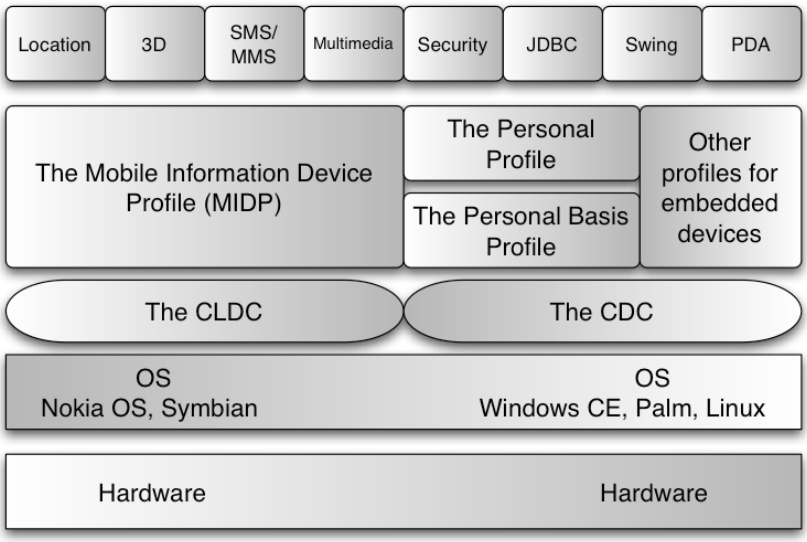

FIGURA 3.9 Arquitectura de la plataforma J2ME. Fuente http://java.sun.com

- **Configuraciones.** Son especificaciones que detallan una máquina virtual y un set básico de APIs que nos permite el trabajo con una clase de dispositivo. Por ejemplo, una configuración podría ser diseñada para dispositivos que cuenten con menos de 512 KB de memoria, la configuración CLDC (*Connected Limited Device Configuration*) 1.0, CLDC 1.1.

- **Perfiles.** Estos añaden más funcionalidad a una configuración. Como anteriormente se mencionó, una configuración describe una JVM y un set de APIs, pero frecuentemente estos no cubren todas las posibles funcionalidades que un dispositivo puede ofrecer, es aquí donde los perfiles cubren esta deficiencia. Por ejemplo, los perfiles comúnmente incluyen APIs para controlar el ciclo del aplicativo, la interfaz de usuario, el almacenamiento de información, etc. Los perfiles mas usados actualmente por dispositivos móviles son el MIDP (*Mobile Information Device Profile*) 1.0 y el MIDP 2.0

- **Paquetes Opcionales.** Proveen funcionalidad adicional que no se encuentra cubierta por una configuración o un perfil especifico. Unos ejemplos de paquetes opcionales son el JDBC

API, que nos provee interfaces para poder realizar consultas con base de datos.

3.2.1.4. Stacks

Un *stack* (pila) esta conformado por una configuración, un perfil y varios o ningún paquete opcional. Un dispositivo trabaja para un *stack* definido, esto significa que este grupo nos asegura el cumplimiento de todas las funcionalidades dadas por el fabricante.

Los primeros celulares con soporte J2ME utilizaban el stack formado por la configuración CLDC 1.0 con el perfil MIDP 1.0, mientras que en la actualidad, comúnmente utilizan el stack formado por la configuración CLDC 1.1, el perfil MIDP 2.0 y los paquetes opcionales SMS/MMS (mensajes de texto o mensajes multimedia) y MMAPI (multimedia).

3.2.2. BREW

BREW es una plataforma de desarrollo de aplicativos para dispositivos móviles que utilizan chips de comunicación fabricados por la empresa QUALCOMM.

En la actualidad soporta los estándares de comunicación GSM/GPRS, UMTS y CDMA (desde sus inicios).

3.2.2.1. Características:

Las aplicaciones para esta plataforma deben ser fácilmente descargables, para ahorrar la memoria del dispositivo. Estas aplicaciones deben estar disponibles sin necesidad de volver a pagar por ellas.

El API de BREW es más estándar sobre los dispositivos que lo soportan, no como en J2ME que tiende a tener variaciones según los dispositivos.

Al utilizar en BREW el lenguaje de programación C++, se suelen utilizar algoritmos que permiten ahorrar una mayor cantidad de memoria durante la ejecución de una aplicación. Esto gracias a la posibilidad de liberar la memoria reservada para cada estructura de almacenamiento y no recurrir a un proceso externo al desarrollo (recolector de basura en JAVA).

3.2.2.2. Arquitectura

El ambiente de ejecución de BREW se ejecuta entre la aplicación y el sistema operativo del dispositivo.

El emulador de BREW (BREW Simulator) no emula el hardware del equipo. En su lugar, la aplicación es compilada y enlazada con librerías propias del BREW que corren bajo el sistema operativo establecido.

3.2.2.3. Proceso de autorización

A diferencia de la plataforma J2ME anteriormente explicada, en BREW un desarrollador no puede descargar una aplicación en algún equipo que la soporte y comenzar a ejecutarla. Es necesario que el software se encuentre digitalmente autorizado por QUALCOMM para esto.

Este proceso es rígido y complejo debido a que BREW en sus APIs proporciona un completo control sobre el hardware del dispositivo incrementando el peligro de software malicioso que dañe permanentemente a este.

Este proceso comienza una vez que la aplicación ha sido terminada y probada internamente por sus desarrolladores. En ese estado debe ser enviada a QUALCOMM para su verificación. Luego de que la aplicación pase las numerosas pruebas de calidad de QUALCOMM, esta será ofrecida a una compañía operadora y se encontrara accesible para que sea descargada en todos los equipos.

3.3. Conclusiones

Para el desarrollo de este trabajo de tesis se ha escogido usar la plataforma J2ME por las siguientes razones:

- Mayor facilidad en las pruebas con los dispositivos. Como se explicó con anterioridad, para poder descargar una aplicación desarrollada a un dispositivo BREW, se debe seguir un complicado proceso, mientras que para el caso del J2ME esto no es necesario.

- Según el cuadro 3.2, el mercado de teléfonos móviles que utilizan el estándar GSM tiene una mayor tasa de aumento que los de tecnología BREW. Las empresas Claro y Nextel utiliza exclusivamente teléfonos móviles que utilizan J2ME mientras que Telefónica cuenta con teléfonos móviles que utilizan BREW, pero también que utilizan J2ME, por lo que en el mercado se impone el uso de teléfonos compatibles con J2ME.

	2000	2001	2002	2003	2004	2005	2006
Telefónica	898,173	1,087,152	1,239,056	1,506,637	2,124,776	3,383,835	5,058,497
Comunicaciones Móviles	373,091	430,282	550,162	650,617	680,493	-	-
Nextel	68,403	110,248	129,780	146,971	184,895	249,475	345,354
Claro / TIM	-	165,602	387,945	626,118	1,102,394	1,950,046	3,368,628
Total Perú	1,339,667	1,793,284	2,306,943	2,930,343	4,092,558	5,583,356	8,772,479

CUADRO 3.2 Líneas por empresa. Fuente: http://www.osiptel.gob.pe

- Mayor variedad de herramientas libres de licencias disponibles en el mercado. La mayoría de entornos de programación cuentan con módulos que facilitan el desarrollo de aplicativos para dispositivos móviles. Esto permite una mejor elección de la herramienta que más se adopte a nuestras necesidades.

4. Requerimientos del software

4.1. Módulos del sistema

Tomando en cuenta los distintos usuarios con los que el sistema cuenta y la funcionalidad deseada, éste se organiza de la siguiente manera:

- **Sistema de Gestión Central (SGC)**

  Módulo encargado de la administración y gestión del servicio de taxi.

  - **Seguridad**

    Se encarga de la seguridad del SGC. Solamente los usuarios autorizados podrán hacer uso de las funcionalidades de este submódulo como son el poder definir determinados perfiles por empresa, la administración de empresas, la administración de usuarios y la autenticación de usuarios entrantes en el sistema.

  - **Configuración del servicio**

Se encarga de la configuración de las variables participantes en el desarrollo de las operaciones relativas al servicio de taxi como son el tipo de automóvil, las zonas, las marcas y poder inscribir nuevas unidades de servicio.

- **Servicio de transporte**

  Se encarga de la gestión de los procesos involucrados en el servicio de taxi como son la toma de pedidos, el registro de la oferta, el seguimiento al servicio, la cancelación del servicio y la finalización de este.

  Además, este submódulo permite contar con información relativa a la unidad de servicio como son su ubicación geográfica, disponibilidad, registro de incidentes, etc.

- **Reportes**

  Se encarga de la generación de reportes sobre el desarrollo de las operaciones de una determinada empresa en el sistema.

- **Aplicación Web / Móvil**

  Se enfoca principalmente en la generación de pedidos por parte de los clientes. Permite crear un pedido, eliminar un pedido, elegir una propuesta, visualizar información del servicio, etc.

  Además se puede visualizar los últimos servicios realizados así como los costos incurridos en estos.

4.2. Requerimientos funcionales

Los requerimientos funcionales definen las acciones fundamentales que el software debe realizar, aceptando y procesando las entradas (*inputs*) y procesando y generando las salidas (*outputs*). [IEEE01]

A continuación se listan los principales requerimientos funcionales con los que la aplicación cuenta.

4.2.1. Sistema de Gestión Central

4.2.1.1. Módulo de Seguridad

- **S1.** Permitir la autentificación de usuarios según nivel de seguridad.
- **S2.** Mantenimiento de empresas.
- **S3.** Mantenimiento de funcionalidades.
- **S4.** Mantenimiento de perfiles.
- **S5.** Permitir asignar accesos a las funcionalidades por perfil.
- **S6.** Mantenimiento de usuarios.
- **S7.** Permitir asignar perfiles a cada usuario.

4.2.1.2. Módulo de Configuración del servicio

- **C1.** Mantenimiento de tipos de unidades de servicio.
- **C2.** Mantenimiento de marcas de vehículo.
- **C3.** Permitir dar de alta y baja una unidad de servicio.
- **C4.** Mantenimiento de zonas.

4.2.1.3. Módulo de Servicio de Transporte

- **E1.** Permitir dar de alta y baja a un chofer.
- **E2.** Permitir la asignación de un chofer a una unidad de servicio.
- **E3.** Permitir listar los pedidos generados por los clientes, priorizándolos por fecha y hora del requerimiento.
- **E4.** Permitir elegir un pedido que se quiera servir y hacer la oferta correspondiente llenando la información respectiva para enviárselo vía SMS (*Short Message Service*) al cliente.
- **E5.** Permitir asignar un pedido a un taxista para que lo atienda.
- **E6.** Permitir recepcionar la respuesta del cliente. Confirmar la respuesta generando un servicio.
- **E7.** Permitir cancelar un pedido por parte de un cliente si es que no se ha sobrepasado el tiempo limite determinado. Se deberá enviar un SMS (*Short Message Service*) a cada cliente aceptando/notificando la cancelación de un pedido/servicio.
- **E8.** Permitir registrar un imprevisto. Se llama Imprevisto a un choque, robo o cualquier desperfecto del carro.

4.2.1.4. Módulo de Reportes

- **R1.** Permitir mostrar reporte de cantidad de servicios por empresa, zona, unidad de servicio y por chofer con total de ingresos por atenciones.
- **R2.** Permitir mostrar información sobre tarifas promedio del mercado (mínimo y máximo) por empresa y por zonas.

- **R3.** Permitir mostrar un reporte de pedidos aceptados y rechazados por tipo de vehículo y por marca.
- **R4.** Permitir mostrar un reporte de ofertas que no fueron aceptadas por cliente, por empresa y/o chofer.
- **R5.** Permitir mostrar un reporte de imprevistos por empresa.
- **R6.** Permitir mostrar un reporte con los siguientes indicadores de calidad: de ofertas enviadas por pedidos recibidos, servicios por ofertas enviadas y servicios por pedidos recibidos agrupados por empresa y chofer.
- **R7.** Permitir generar reporte de choferes y unidades servicios asignados por fecha y a la fecha.
- **R8.** Permitir mostrar los siguientes tipos de ranking: ranking de las empresas que más reciben aceptación de pedido y ranking de los choferes que más reciben aceptación de pedidos por empresa.

4.2.2. Aplicativo Móvil

4.2.2.1. Módulo del Cliente

- **M1.** Permitir ingresar un pedido de cliente. Se especificará dirección de origen, dirección destino, tipo de automóvil, fecha y hora de recojo.
- **M2.** Permitir la visualización de la Ofertas dadas por las empresas de taxi. Los datos que se visualizan son el nombre de la compañía, tiempo aproximado y tarifa.

- **M3.** Permitir elegir la propuesta que se desee.
- **M4.** Permitir la visualización de todos los servicios que ha pedido un cliente. Además se podrá ver la información relativa al servicio (información del chofer, hora aproximada de llegada) así como su estado.
- **M5.** Permitir cancelar un pedido si es que no ha transcurrido el tiempo permitido. Si se sobrepasa el tiempo permitido, es posible la cancelación del servicio con penalización.
- **M6.** Permitir cancelar un servicio si es que no ha transcurrido el tiempo permitido.
- **M7.** Permitir la visualización de los últimos servicios realizados históricamente así como el total de gastos incurridos en el servicio.

4.3. Requerimientos no funcionales

Los siguientes requerimientos no funcionales permiten el eficiente uso de la aplicación desarrollada en el ámbito de los procesos del negocio de taxis:

- Desarrollo del aplicativo parte móvil utilizando la plataforma J2ME específicamente para dispositivos móviles de la marca Motorota iDEN que soporten MIDP2.0 y CLDC 1.0.
- Desarrollo del SGC (Sistema de Gestión Central) sobre una plataforma J2EE.
- El sistema administrador de base de datos a utilizar es el PostgreSQL versión 8.1.
- Utilizar el servidor de aplicaciones Tomcat versión 5.5.

- Los tiempos de transacciones para la comunicación entre el dispositivo móvil y el SGC deberá ser menor a los 20 segundos.

5. Análisis del software

5.1. Especificación de actores del sistema

Un actor es la abstracción de entidades externas a un sistema que interactúan con este. Estas entidades pueden ser un usuario, un conjunto de usuarios, un subsistema u otro sistema. [UML01]

A continuación se listan los actores participantes en la aplicación.

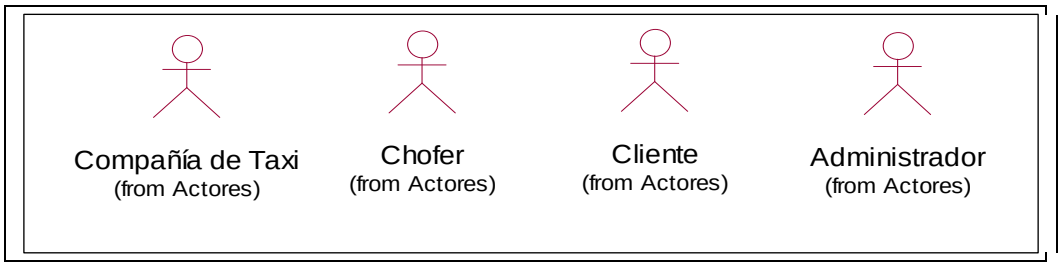

FIGURA 5.1 Actores participantes del sistema

Administrador

Actor que interactúa con el SGC (Sistema de Gestión Central). Tiene la posibilidad de configurar la seguridad a nivel de empresas (creación de perfiles, funcionalidades y relación entre estos) además de dar de alta en el sistema a las empresas que se requieran. Este actor al ser exclusivo del aplicativo, no refiere a ningún rol de la sección 2.2.1 (Participantes en el negocio).

Compañía de Taxi

Actor que trabaja para la compañía de taxi. Este actor tendrá un perfil asignado que le permitirá acceder a las funcionalidades del sistema. También administrará y gestionará los procesos relacionados con el servicio de taxi como son los concernientes a los choferes, pedidos, ofertas y servicios. Refiere al rol Compañía de Taxi de la sección 2.2.1 (Participantes en el negocio).

Chofer

Este actor representa al chofer de taxi que trabaja para una compañía de taxi. Refiere a los roles de Chofer Independiente y Chofer de Compañía de la sección 2.2.1 (Participantes en el negocio).

Cliente

Este actor representa al usuario principal del servicio de taxi que es el principal cliente de las empresas de taxi. Refiere al rol Cliente de la sección 2.2.1 (Participantes en el negocio).

5.2. Especificación de casos de uso

Un caso de uso es una unidad coherente de funcionalidad que proporciona un clasificador (un sistema, subsistema o clase) tal como lo manifiestan las secuencias de mensajes que se intercambian entre el sistema y uno o más usuarios externos (que se representan como actores). [UML01]

Para permitir una mejor comprensión, los casos de uso se organizarán por paquetes (agrupación de elementos y diagramas del modelo [UML01]).

5.2.1. Paquete de Seguridad (Sistema de Gestión Central)

A continuación se especificarán los casos de uso que pertenecen al paquete de Seguridad, como se refiere en la Figura 5.2.

5.2.1.1. Caso de uso Autentificación de usuarios

- **Requerimiento**

  Autentificación de usuarios según nivel de seguridad (S1).

- **Actores involucrados**

  Compañía de Taxi, Administrador

- **Descripción**

  El actor ingresará su nombre de usuario y contraseña para que el sistema permita o no su entrada y le asigne la funcionalidad a la que puede acceder según su nivel de seguridad (perfiles que tiene).

5.2.1.2. Caso de uso Mantenimiento de empresas

- **Requerimiento**

  Mantenimiento de empresas (S2).

- **Actores involucrados**

Administrador

- **Descripción**

  El actor podrá adicionar, modificar o eliminar una empresa de taxi del sistema.

5.2.1.3. Caso de uso Mantenimiento de funcionalidades

- **Requerimiento**

  Mantenimiento de funcionalidades (S3).

- **Actores involucrados**

  Administrador

- **Descripción**

  El actor podrá adicionar, modificar o eliminar una funcionalidad del sistema.

5.2.1.4. Caso de uso Mantenimiento de perfiles

- **Requerimiento**

  Mantenimiento de perfiles (S4). Permitir asignar accesos a las funcionalidades por perfil (S5).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

El actor podrá adicionar, modificar o eliminar perfiles del sistema. Cada perfil consta de una o más funcionalidades que serán utilizadas para la autentificación de usuarios.

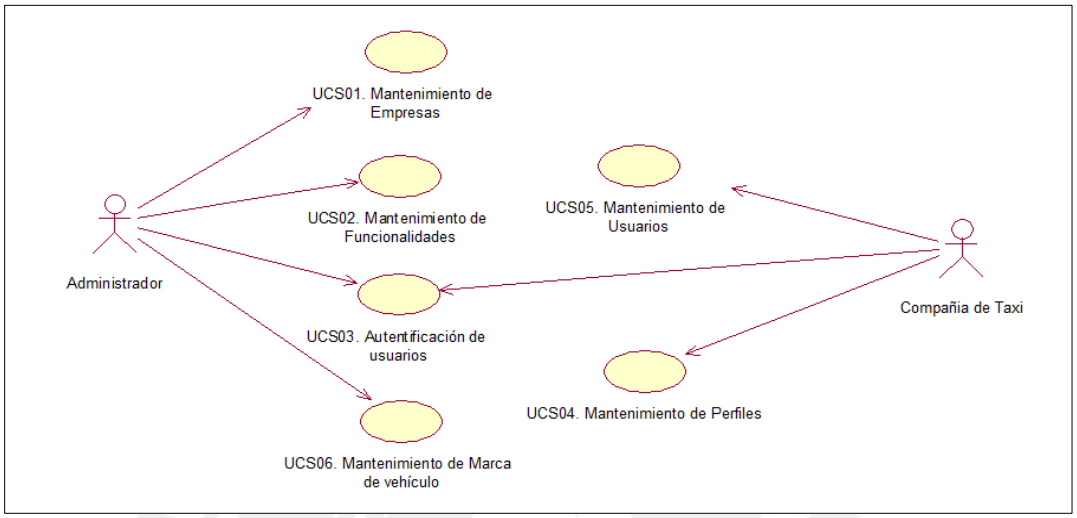

FIGURA 5.2 Casos de uso Módulo de Seguridad

5.2.1.5. Caso de uso Mantenimiento de usuarios

- **Requerimiento**

  Mantenimiento de usuarios (S6). Permitir asignar perfiles a cada usuario (S7).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  El actor podrá adicionar, modificar o eliminar usuarios del sistema. Cada usuario cuenta con uno o más perfiles que le permitirá acceder a diversas partes del sistema.

5.2.1.6. Caso de uso Mantenimiento de marca de vehículo

- **Requerimiento**

  Mantenimiento de marca de vehículo (C2).

- **Actores involucrados**

  Administrador

- **Descripción**

  El actor podrá registrar las diferentes marcas de los vehículos con que contará.

5.2.2. Paquete de Configuración del Servicio (Sistema de Gestión Central)

A continuación se especificarán los casos de uso que pertenecen al paquete de Configuración del Servicio, como se refiere en la Figura 5.3.

5.2.2.1. Caso de uso Mantenimiento de tipo de unidades

- **Requerimiento**

  Mantenimiento de tipo de unidades (C1).

- **Actores involucrados**

  Administrador

- **Descripción**

El actor podrá adicionar, modificar o eliminar un tipo de unidad de servicio del sistema.

5.2.2.2. Caso de uso Mantenimiento de zonas

- **Requerimiento**

  Mantenimiento de zonas (C4).

- **Actores involucrados**

  Administrador

- **Descripción**

  El actor podrá adicionar, modificar o eliminar una zona del sistema.

5.2.2.3. Caso de uso Mantenimiento de unidades de servicio

- **Requerimiento**

  Dar de alta una unidad de servicio, dar de baja una unidad de servicio (C3).

- **Actores involucrados**

  Compañía de taxi

- **Descripción**

  El actor podrá adicionar, actualizar o dar de baja unidades de servicio (vehículos) pertenecientes a una empresa de taxi. El usuario deberá de especificar los datos del vehículo (placa, marca, año, color y número de asientos).

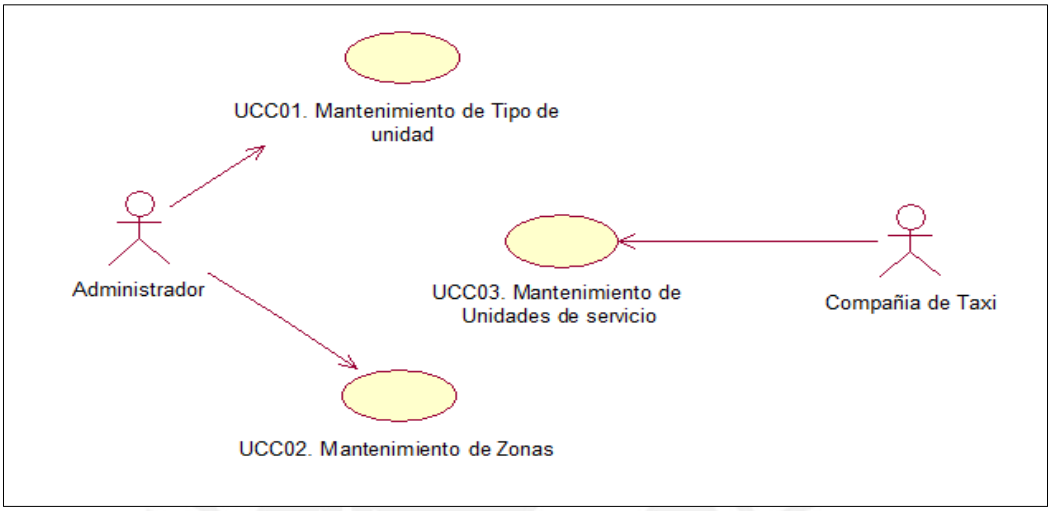

FIGURA 5.3 Casos de uso Módulo de Configuración del servicio

5.2.3. Paquete Servicio de Transporte (Sistema de Gestión Central)

A continuación se especificarán los casos de uso que pertenecen al paquete de Servicio de Transporte, como se refiere en la Figura 5.4.

5.2.3.1. Caso de uso Dar de alta a un chofer

- **Requerimiento**

  Dar de alta a un chofer (E1).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

En este caso de uso se inscribirá un chofer que luego podrá ser asignado a una unidad de servicio. Para el caso que un chofer venga con su vehículo propio, se dará la posibilidad también de registrar una Unidad de servicio realizando el caso de uso del punto 5.2.2.3 (Caso de uso mantenimiento de unidades de servicio).

5.2.3.2. Caso de uso Dar de baja a un chofer

- **Requerimiento**

  Dar de baja a un chofer (E1).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se pondrá fuera de servicio a un chofer que no podrá ser asignado a ninguna unidad de servicio. Cabe resaltar que en este caso de uso se inhabilita permanentemente del sistema al chofer y no solamente por un tiempo determinado.

5.2.3.3. Caso de uso Responder pedido de cliente

- **Requerimiento**

  Listar los pedidos que se tienen que servir por un lapso de tiempo determinado (E3) y elegir un pedido que se quiera servir y hacer la oferta correspondiente notificando vía SMS (*Short Message Service*) al cliente (E4).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso en un primer momento listará los pedidos que han sido dados por los clientes. De estas, el actor elegirá la que más le conviene para poder ofrecer una oferta de servicio.

5.2.3.4. Caso de uso Generación de servicio

- **Requerimiento**

  Asignar un pedido a un chofer para que lo atienda (E5) y recepcionar la respuesta a la oferta dada por el cliente generando un servicio con su respectivo chofer asignado (E6).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se elegirá una oferta respondida por el cliente y se generará un servicio asignando a una unidad de servicio con su respectivo chofer asignado.

5.2.3.5. Caso de uso Asignación de choferes a unidades de servicio

- **Requerimiento**

  Asignación de un chofer a una unidad de servicio (E3).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se asignará un chofer a una unidad de servicio para un rango de fecha determinado y un turno determinado (Turno mañana, tarde, noche) y para los días determinados.

5.2.3.6. Caso de uso Cancelación de servicio

- **Requerimiento**

  Cancelar un servicio de un cliente si es que se ha sobrepasado el tiempo límite determinado, se notificará vía un SMS (*Short Message Service*) a un cliente notificando la cancelación de su servicio (E7).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se podrá cancelar un servicio de un cliente notificándole vía SMS (*Short Message Service*).

5.2.3.7. Caso de uso Registro de imprevistos

- **Requerimiento**

  Permitir registrar imprevisto. Se llama Imprevisto a un choque, robo o cualquier desperfecto del carro (E8).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se registrará un imprevisto en el cual se vió involucrada una unidad de servicio.

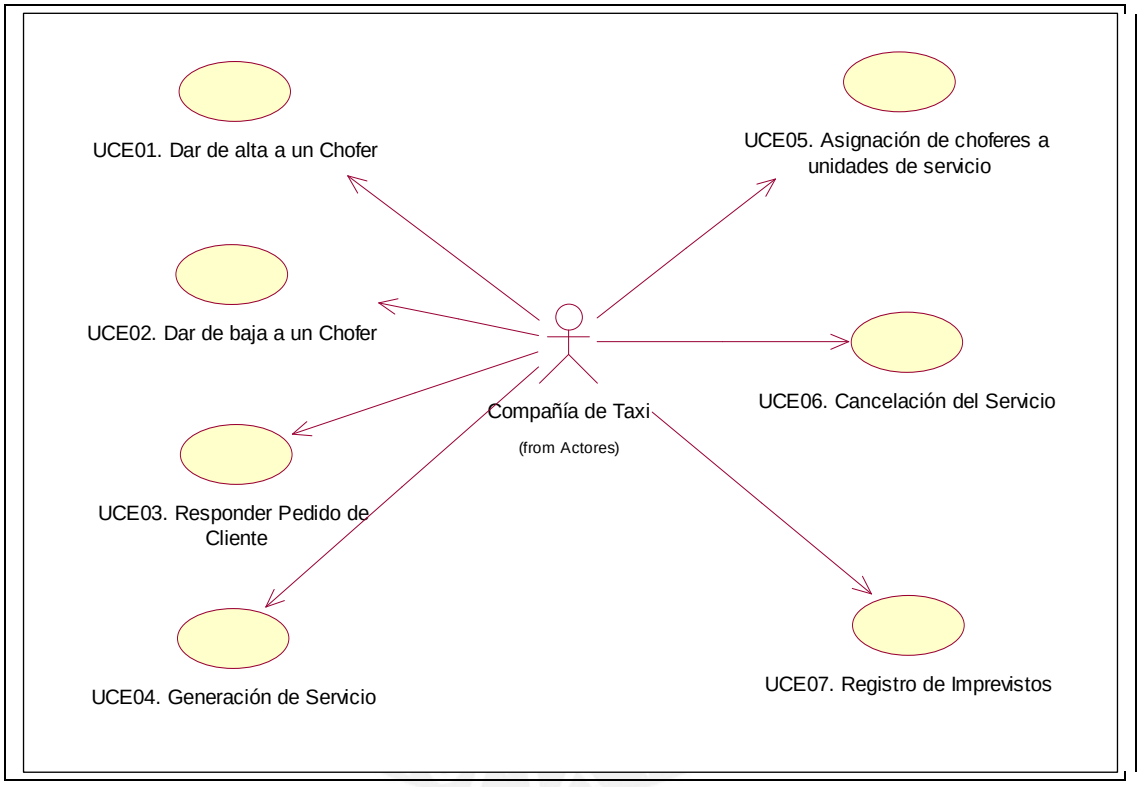

FIGURA 5.4 Casos de uso Módulo del Servicio de transporte

5.2.4. Paquete de Reportes (Sistema de Gestión Central)

A continuación se especificarán los casos de uso que pertenecen al paquete de Reportes, como se refiere en la Figura 5.5.

5.2.4.1. Caso de uso Reportes de servicio

- **Requerimiento**

  Permitir mostrar reporte de cantidad de servicios de la empresa, zona, unidad de servicio y por chofer con total de ingresos por atenciones (R1).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  En este caso de uso se permitirá a los usuarios del tipo compañía de taxi generar reportes con la cantidad de servicios ingresados, atendidos y la relación entre estas cantidades.

5.2.4.2. Caso de uso Reportes de tarifa

- **Requerimiento**

  Permitir mostrar información sobre tarifas promedio del mercado (mínimo y máximo) de la empresa y por zonas (R2).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso permitirá a los usuarios del tipo compañía de taxi generar reportes indicando los montos promedio, mínimo y máximo de las tarifas de servicio por zonas. En este reporte además se indicará los montos extras (bonificaciones o

descuentos) facturados a los clientes por situaciones inesperadas (tardanzas, cambios de ruta, etc).

5.2.4.3. Caso de uso Reportes de pedido

- **Requerimiento**

  Permitir mostrar un reporte de pedidos aceptados y rechazados por tipo de vehículo y por marca (R3).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso permitirá al usuario compañía de taxi generar un reporte que indique las cantidades de pedidos que son aceptados y rechazados por los clientes indicando el tipo de vehículo y marca que fueron asignados. Esto permitirá a la empresa conocer las preferencias de los clientes en cuanto a la calidad del servicio.

5.2.4.4. Caso de uso Reportes de propuesta

- **Requerimiento**

  Permitir mostrar un reporte de ofertas que no fueron aceptadas por cliente y/o chofer (R4).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

Este caso de uso permitirá a los usuarios empresa de taxi conocer cuales son las cantidades de ofertas de una empresa que fueron rechazadas. El conteo se efectuará por cliente, por empresa y por chofer, dependiendo de lo que requiera el usuario.

5.2.4.5. Caso de uso Reportes de imprevistos

- **Requerimiento**

  Permitir mostrar un reporte de imprevistos de la empresa (R5).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso permitirá a un usuario empresa de taxi generar un reporte con los imprevistos que ha tenido en las atenciones de servicios.

5.2.4.6. Caso de uso Reportes de calidad

- **Requerimiento**

  Permitir mostrar un reporte con los siguientes indicadores de calidad: ofertas enviadas por pedidos recibidos, servicios por ofertas enviadas y servicios por pedidos recibidos agrupados por chofer (R6).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso permitirá a los usuarios de la empresa de taxi generar reportes con indicadores de calidad que permitan conocer: número de ofertas enviadas por pedidos recibidos, número de servicios por ofertas enviadas y número de servicios por pedidos recibidos agrupados por chofer.

5.2.4.7. Caso de uso Reporte de unidades de servicio

- **Requerimiento**

  Permitir generar reporte de choferes y unidades servicios asignados por fecha y a la fecha (R7).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Este caso de uso permitirá a los usuarios compañía de taxi generar reportes con la historia de asignación de choferes a unidades de servicio por fecha de asignación y a la fecha actual.

5.2.4.8. Caso de uso Generar rankings

- **Requerimiento**

  Permitir mostrar los siguientes tipos de ranking: ranking de las empresas que más reciben aceptación de pedido y ranking de los choferes que más reciben aceptación de pedidos (R8).

- **Actores involucrados**

  Compañía de Taxi

- **Descripción**

  Permite a los usuarios compañía de taxi generar ranking que les permita conocer cuales son las empresas que tienen la mayor cantidad de aceptación de pedidos y cuales son los choferes con mayor aceptación de pedidos por los clientes.

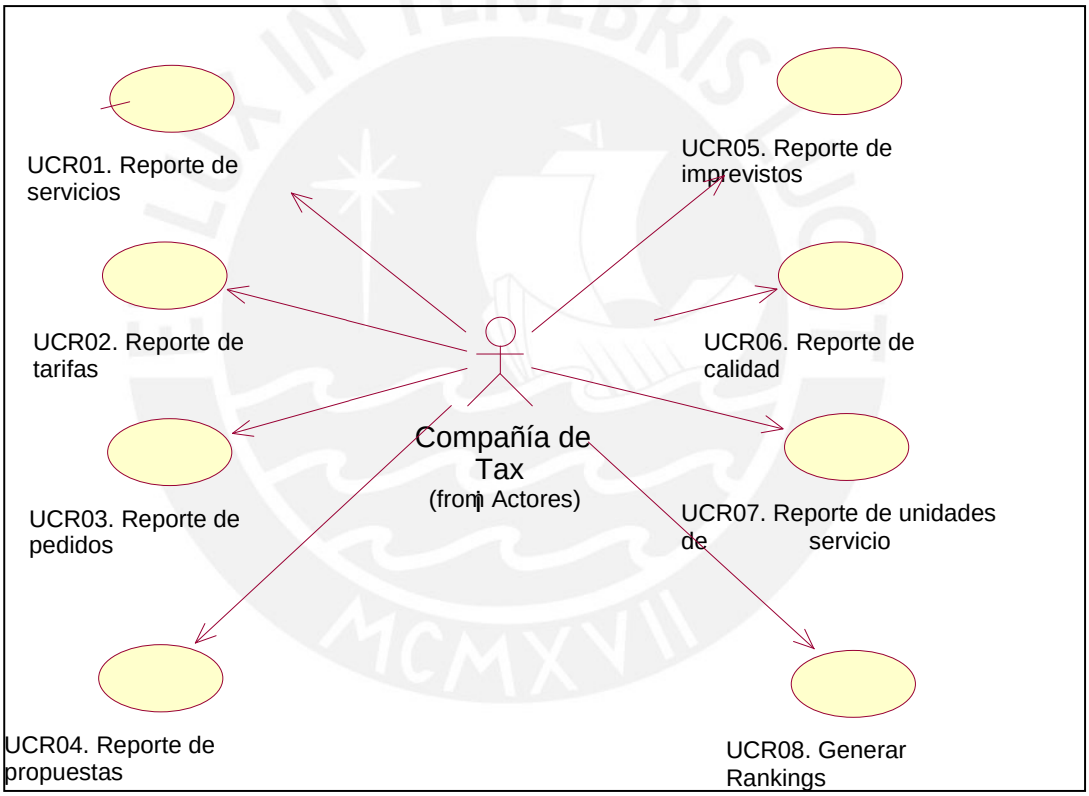

FIGURA 5.5 Casos de uso Módulo de Reportes

5.2.5. Paquete Módulo Cliente (Aplicativo Web / Móvil)

A continuación se especificarán los casos de uso que pertenecen al paquete Módulo Cliente, como se refiere en la Figura 5.6.

5.2.5.1. Caso de uso Ingreso de pedidos

- **Requerimiento**

  Permitir ingresar un pedido de servicio. Se especificará dirección de origen, dirección destino, tipo de automóvil, fecha y hora de recojo (M1).

- **Actores involucrados**

  Cliente

- **Descripción**

  En este caso de uso el actor ingresará los detalles de un pedido de servicio de taxi. Se especificará las características del servicio que el cliente requiere (dirección de origen, dirección destino, tipo de automóvil, fecha y hora de recojo).

5.2.5.2. Caso de uso Elección de propuestas de pedidos

- **Requerimiento**

  Permitir la visualización de las ofertas del pedido dadas por las empresas de taxi. Los datos que se visualizan son el nombre de la compañía, tiempo aproximado y tarifa (M2). Permitir elegir la oferta que se desee (M3).

- **Actores involucrados**

  Cliente

- **Descripción**

En este caso de uso se podrán visualizar las ofertas respondidas por parte de las compañías de taxi. Posteriormente, el actor (cliente) podrá elegir si le satisface o no alguna.

5.2.5.3. Caso de uso Cancelación de pedido

- **Requerimiento**

  Permitir cancelar un pedido (M5).

- **Actores involucrados**

  Cliente

- **Descripción**

  En este caso de uso, se mostrarán los pedidos realizados por el cliente que no han generado un servicio aun. El actor podrá elegir uno y cancelarlo para que no aparezca en su bandeja de entrada de pedidos.

5.2.5.4. Caso de uso Visualización de servicio

- **Requerimiento**

  Permitir la visualización de todos los servicios que ha generado un cliente. Además se podrá ver la información relativa al servicio (información del chofer, hora aproximada de llegada) así como su estado (M4).

- **Actores involucrados**

  Cliente

- **Descripción**

En este caso de uso se visualizará una lista de servicios que se encuentran en curso, pudiendo el actor seleccionar uno de estos para ver los detalles de este: información del chofer, hora aproximada de llegada y estado del servicio (si el chofer se encuentra en camino a recoger al cliente, si ya llegó al lugar de recojo y si ya finalizó el servicio).

5.2.5.5. Caso de uso Visualización de últimos servicios realizados

- **Requerimiento**

  Permitir la visualización de los últimos servicios realizados históricamente así como el total de gastos incurridos en el servicio (M7).

- **Actores involucrados**

  Cliente

- **Descripción**

  En este caso de uso se visualizarán los 10 últimos servicios realizados por el cliente.

5.2.5.6. Caso de uso Cancelación de un servicio

- **Requerimiento**

  Permitir la cancelación de un servicio. Para esto se deberá verificar que no se haya pasado un tiempo determinado (M6).

- **Actores involucrados**

Cliente

- **Descripción**

  En este caso de uso se podrá cancelar un servicio en curso. Viene incluido en el caso de uso Visualización de servicio (sección 5.2.5.4)

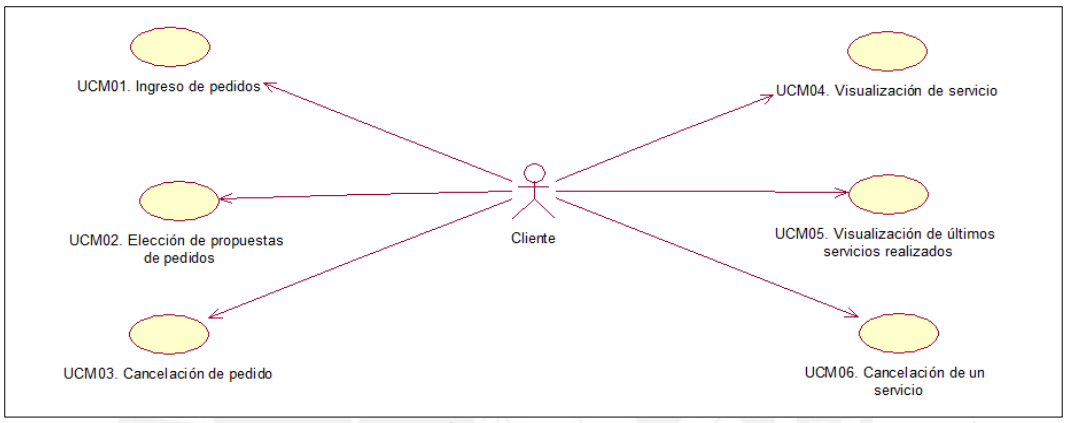

FIGURA 5.6 Casos de uso Módulo de Clientes del aplicativo Web / móvil

5.3. Clases de análisis

Un diagrama de clases es una presentación gráfica de la vista estática, que muestra una colección de elementos declarativos (estáticos) del modelo, como clases, tipos, sus contenidos y relaciones. [UML01]

Las clases de análisis de la aplicación han sido elaboradas según el estándar UML y se han agrupado en paquetes en forma similar a los casos de uso para su mayor facilidad de manejo y entendimiento. A continuación se detallan estos paquetes.

5.3.1. Paquete Seguridad

Este paquete contiene las clases pertenecientes al módulo de seguridad. El diagrama correspondiente a este paquete se muestra en la figura 5.7.

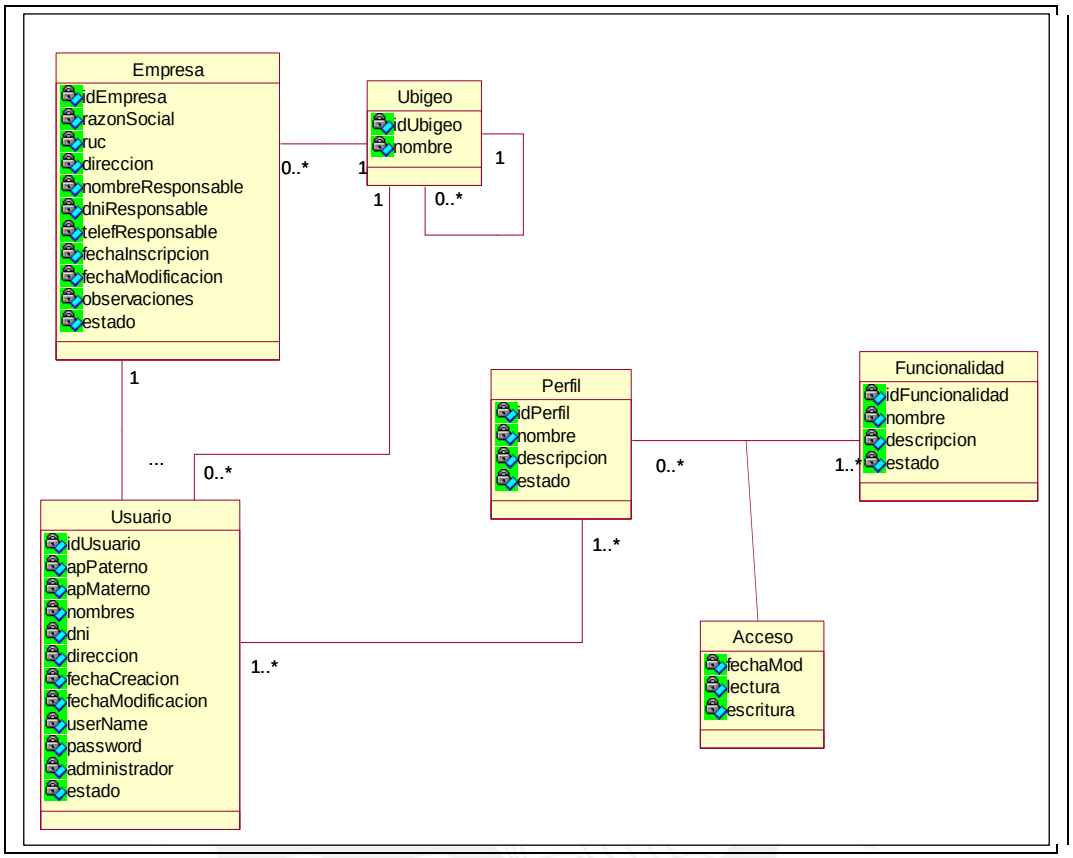

FIGURA 5.7 Diagrama de clases Paquete Seguridad

- **Empresa**

  Clase que conceptualiza al actor Compañía de Taxi. Guarda información relativa a él. Para el alcance de esta tesis, solamente se tendrá un objeto empresa.

- **Ubigeo**

  Clase que representa un ubigeo. El ubigeo es un tipo de codificación geográfica avalada por el INEI (Instituto Nacional de Estadística e Informática) que nos servirá para situar geográficamente las diferentes entidades participantes en el sistema.

- **Usuario**

Clase que representa al actor Usuario. El usuario pertenece a una sola empresa y se le asigna un nombre de usuario y una contraseña para que de esa manera pueda acceder al sistema. Además, pueden ser estos administradores con la habilidad de inscribir a otros usuarios.

- **Perfil**

  Clase que representa un perfil dentro del sistema. Este perfil englobará un grupo de funcionalidades las cuales el usuario que tenga dicho perfil, podrá acceder.

- **Funcionalidad**

  Clase que representa al concepto de Funcionalidad dentro del sistema. Este concepto permite el acceso de cada usuario a ciertas partes del sistema.

- **Acceso**

  Clase que representa los permisos que tendrá una Funcionalidad dado un perfil. Estos permisos pueden ser de lectura o de lectura-escritura.

5.3.2. Paquete Configuración del Servicio

Este paquete contiene las clases pertenecientes al módulo de configuración del servicio. El diagrama correspondiente a este paquete se muestra en la figura 5.8.

- **TipoUnidad**

  Clase que representa el tipo de vehículo que es la unidad de servicio.

- **Unidad de Servicio**

  Clase que representa el vehículo de transporte que será utilizado por un chofer. Esta unidad pertenece a una compañía de taxi.

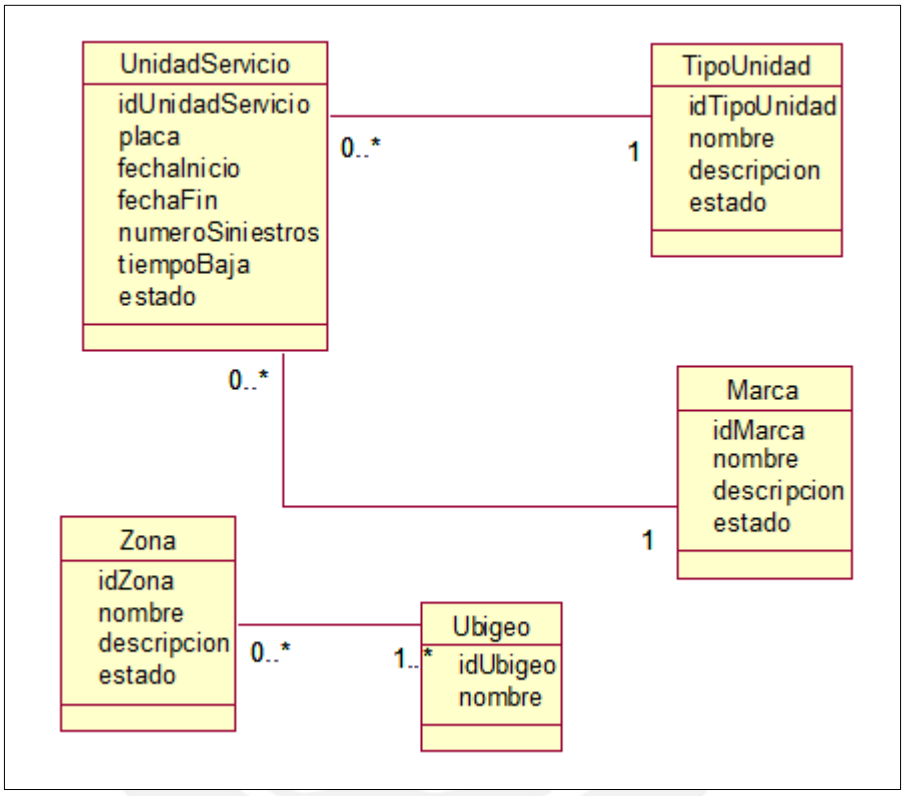

FIGURA 5.8 Diagrama de clases Paquete Configuración del Servicio

- **Marca**

  Clase que representa la marca que es la Unidad de Servicio.

- **Zona**

  Clase que representa un área geográfica de la ciudad. Esta área puede estar formada por uno o más ubigeos.

5.3.3. Paquete Servicio de Transporte

Este paquete contiene las clases pertenecientes al módulo de servicio de transporte. El diagrama correspondiente a este paquete se muestra en la Figura 6.9.

- **Chofer**

  Clase que representa a la persona encargada de realizar los Servicios utilizando la Unidad de Servicio.

- **Moneda**

  Clase que representa el tipo de moneda (dólares o soles) con que el cliente desea pagar el servicio.

- **Pedido**

  Clase que representa un pedido realizado por un cliente vía la aplicación móvil.

- **Propuesta**

  Clase que representa una propuesta de servicio dada por una compañía de taxi.

- **Servicio**

  Clase que representa la acción de un servicio. Este servicio se generará cuando el cliente acepta una de las propuestas dadas por las compañías de taxi.

- **Imprevisto**

  Clase que representa una situación ocurrida en la realización de un servicio. Esto puede ser un robo, siniestro, pistas cerradas, etc.

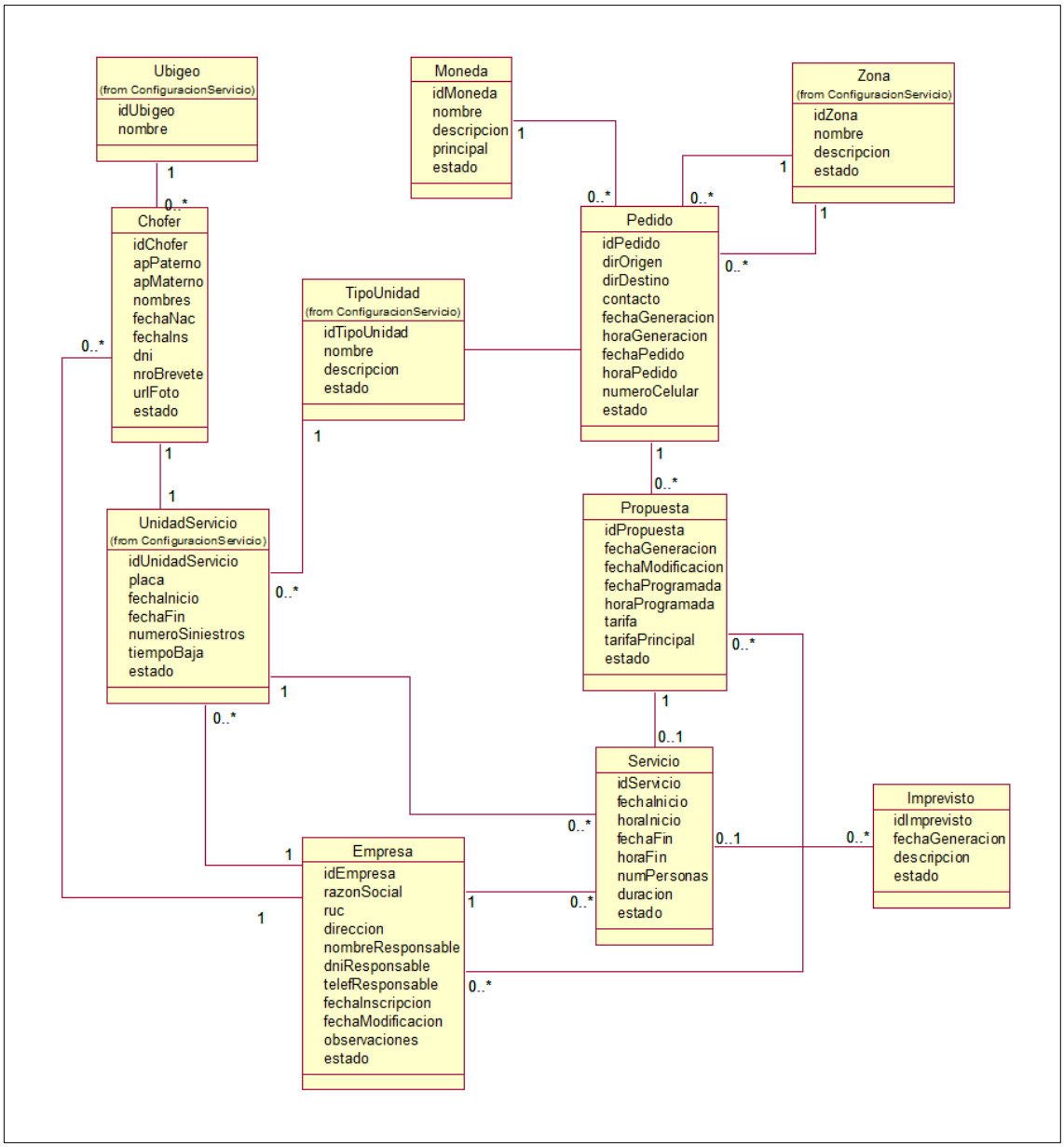

FIGURA 5.9 Diagrama de clases Paquete Servicio de Transporte

6. Diseño del Software

La fase de diseño, dentro de la vida del desarrollo de software, se caracteriza por definir los aspectos técnicos de la solución a desarrollar diferenciándose así de la fase de análisis que se centra en los aspectos funcionales.

Muchos de los diagramas definidos por RUP se utilizan en esta etapa del desarrollo del software.

6.1. Consideraciones iniciales

En esta sección se describen muchos de los puntos que se tomaron en cuenta durante el desarrollo de la etapa de diseño que influirán directamente en el producto final.

6.1.1. Dependencias

6.1.1.1. Software y hardware

Al ser esta una solución que contará con diferentes perfiles de usuarios y estos usuarios interactuarán con la solución de distinta manera y con diferentes dispositivos, es importante plantear los equipos y aplicaciones con que se contará.

- **Aplicación Móvil.** Los dispositivos móviles que en principio podrán soportar la aplicación son los que permiten correr aplicaciones J2ME. Pero para efectos de la implementación de esta solución, se optará por los modelos de la marca Motorota modelos iDEN. El lenguaje de programación elegido para su implementación será el Java.

- **Sistema de Gestión Central.** Se ejecuta bajo el servidor Web llamado Apache Tomcat versión 5.5 así como una base de datos PostgreSQL versión 8. Estas aplicaciones son instaladas bajo un sistema operativo GNU/Linux Debian Etch. El lenguaje de programación elegido para su implementación será el Java.

- **Demonio de recepción y transmisión de comunicaciones.** Se ejecutará bajo un servidor GNU/Linux Debian Etch. El lenguaje de programación elegido será el Java.

6.1.1.2. Interacción del usuario final

Al ser esta una aplicación orientada a usuarios inexpertos, las interfaces gráficas deben ser sencillas y entendibles. Los lineamientos seguidos son:

- Mostrar claramente las opciones y las diferentes formas de acceder a ellas. Así como las opciones que momentáneamente estén inactivas.
- Mostrar al usuario el estado en que se encuentra cualquier proceso que haya realizado, para que así no piense que la aplicación ha entrado en un ciclo infinito sin salida.
- Tener indicadores que permitan regresar a la pantalla anterior cancelando todo lo ingresado hasta ese momento.
- No permitir que se abran más de tres ventanas emergentes.
- Mantener una constante retroalimentación con el usuario sobre el desarrollo de los procesos que generó.
- Contar con un diseño consistente a través de las diferentes pantallas que se presentarán.

6.1.1.3. Restricciones

Como se explicó en la sección 6.1.1.1 (Software y hardware), el universo de dispositivos móviles se reduce a los mencionados anteriormente.

La seguridad se maneja desde la aplicación pero también se contará con distintos perfiles para los accesos a la base de datos.

Dada la capacidad de los dispositivos móviles, el manejo de la memoria en la aplicación móvil es de suma importancia. Se utilizan formas de programación orientadas al ahorro de memoria.

6.2. Arquitectura y Organización

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes permitiendo tener una visión común entre los componentes involucrados y una perspectiva clara del sistema completo [KRU2000].

La metodología RUP además de plantear que el desarrollo de aplicaciones sea guiado por casos de uso, también indica que se debe prestar atención tempranamente a la arquitectura para así evitar que la aplicación sea fuertemente impactada por los cambios posteriores.

6.2.1. Visión de alto nivel

El despliegue de la aplicación será el siguiente:

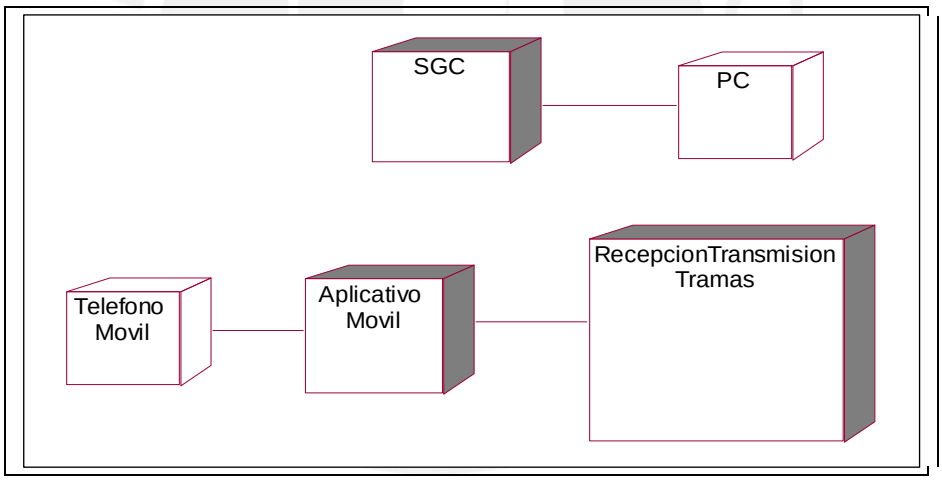

FIGURA 6.10 Arquitectura de alto nivel

Donde **Teléfono Móvil y PC** son los dispositivos que sirven de entrada para los procesos.

Aplicativo Web / Móvil es la aplicación que va en los teléfonos celulares que se encargará del ingreso de pedido y confirmación de estos.

SGC (Sistema de Gestión Central) es la aplicación dirigida a las empresas de taxi para que puedan configurar y gestionar su servicio.

Recepción y Transmisión de Tramas es el proceso que sirve de puente de comunicación entre el aplicativo móvil y la base de datos.

6.2.2. Framework

Un *framework* es un conjunto de clases cooperantes que constituyen un diseño reutilizable para una clase especifica de software [DEU1989].

La elección de un *framework* es una actividad muy importante ya que define la forma de la arquitectura. Dependiendo del tipo de aplicación que se requiera construir, se debe de analizar las facilidades que brindan los diferentes *frameworks* así como las dificultades que traen, eligiendo el que más se acerque a lo requerido.

6.2.3. Struts

Una mala práctica que se solía tener al momento de desarrollar aplicaciones Web, era el no tener delimitado el campo de acción de las distintas clases participantes, incurriendo en altos niveles de acoplamiento y poca legibilidad del código. El mantenimiento de aplicaciones realizadas de esta manera se tornaba muy tedioso y estas dependían en gran medida de los programadores que participaron en el desarrollo siendo casi imposible que terceras personas pudieran realizar esta labor. Con Struts, se delimitó el ámbito de las clases participantes según el tipo de participación que tengan en la aplicación, definiendo de una mejor manera la interacción entre estas, permitiendo código más legible y mantenible.

Struts consiste en un *framework* que facilita el desarrollo de una aplicación Web. Sirve para arquitecturas que utilizan el patrón MVC (*Model View Controller*)

Struts provee su propio componente de Control y se integra con otras tecnologías para definir las demás capas del patrón MVC. Para la capa del Modelo (*Model*), Struts puede interactuar con tecnologías de acceso a datos como son JDBC, EJB, Hibernate por citar algunas.

Por su parte, para la parte de la Vista (*View*), Struts utiliza la tecnología JSP (*Java Server Pages*) que nos muestra código java embebido en una página HTML. Esta tecnología nos permite tener páginas más dinámicas dado que podemos insertar lógica en ellas, además de permitirnos manejar objetos JAVA dentro del contexto de la página Web.

En la capa de Control (*Controller*) es donde Struts tiene mayor influencia. Introduce la utilización de un archivo de configuración de formato XML que es donde se define el flujo de la aplicación.

6.2.3.1. Arquitectura Struts

Struts divide de una manera sencilla y clara las capas del patrón MVC como se aprecia en la Figura 6.2.

- **Vista (*View*)**

  Página JSP.

  Es la página Web codificada en lenguaje HTML con código JAVA incrustado. Acá se codifica la interfaz de usuario con el sistema (ingreso y salida de datos).

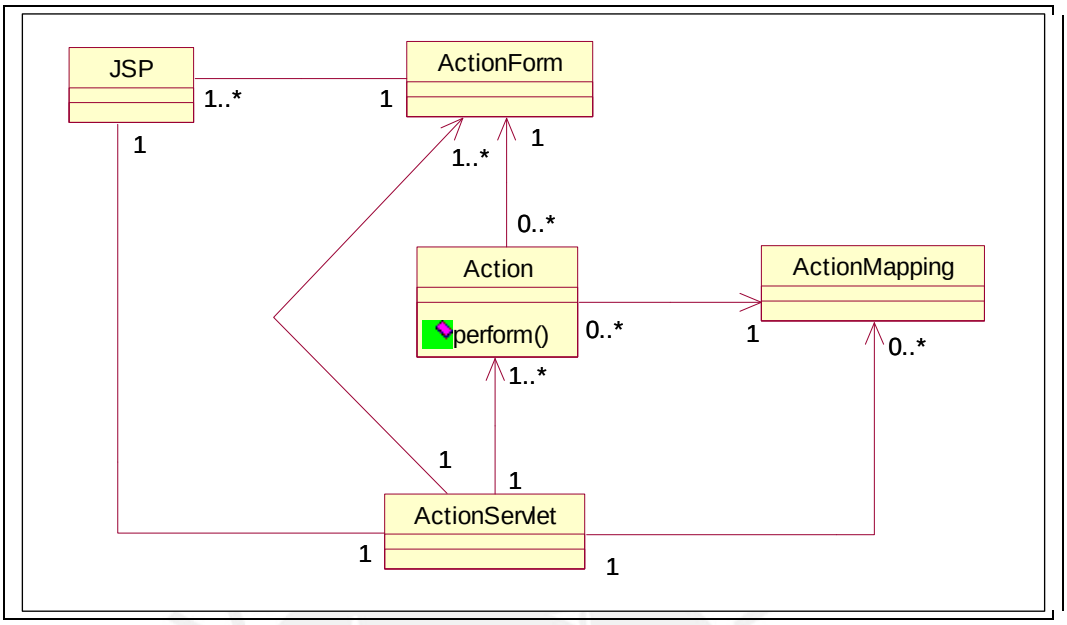

FIGURA 6.11 Clases participantes en Struts

- **Control (*Controller*)**

Clase ActionServlet

Esta clase se encarga de direccionar peticiones HTTP del cliente (para nuestro caso, una página JSP) a las correspondientes clases Action (se explicará posteriormente) decidiendo que función de la lógica del negocio se llevará a cabo además de delegar la responsabilidad de producir la siguiente interfaz de usuario.[RED2002]

Además se encarga de leer la información y crear la clase Action correspondiente definida en el archivo de configuración struts-config.xml (donde se definen los *mappings* que son parámetros que relacionan una URI con una función a realizar).

Clase ActionMapping

Como se explicó con anterioridad, la clase ActionServlet necesita conocer de qué manera cada petición HTTP es generada por el usuario creando la clase Action apropiada. Para

esto, la clase ActionMapping encapsula una URI y la clase Action respectiva.

- **Modelo (*Model*)**

  Clase JavaBean

  Son clases definidas por el desarrollador que implementan lógica del negocio así como la interacción con recursos externos (base de datos, sistemas *legacy*).

Para permitir una comunicación ordenada entre las distintas capas del patrón MVC, Struts define clases de interfaz entre capas.

Clase ActionForm

Esta clase representa a una página Web JSP que sirve como interfaz gráfica de usuario. Cada elemento HTML que permita ingreso o salida de datos por formulario cuenta con su respectivo atributo en una clase del tipo de esta.

La clase ActionServlet usa una instancia de ActionForm como representación de los datos de la página JSP. El uso de esta clase es delegada a la instancia de una clase Action que se genera para la petición requerida.

Clase Action

Esta clase sirve de medio de comunicación entre la capa de Control y la de Modelo,

Es generada por el ActionServlet para poder realizar una función. Para poder realizar esta función, utiliza el ActionForm que el ActionServlet tiene.

Al momento de realizar una función determinada, ésta crea y utiliza las clases pertenecientes a la capa del Modelo para poder realizar la función del negocio.

6.2.4. Hibernate

Hibernate es una herramienta ORM (*Object Relationship Manager*) encargada de lograr persistencia en objetos JAVA.

La base del desarrollo en Hibernate es el uso de objetos POJOs (*Plain Old Java Objects*) para lograr comunicar una aplicación JAVA con una fuente de datos de una manera transparente del tipo de esta.

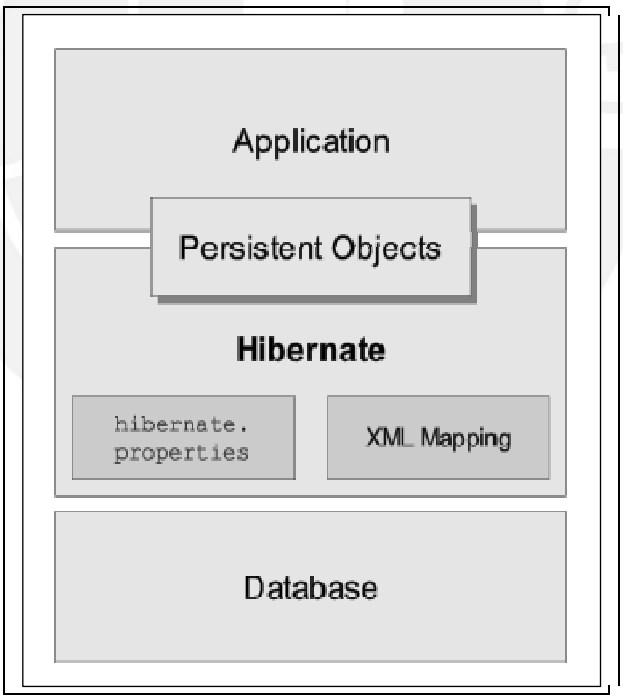

FIGURA 6.12 Ubicación del Hibernate dentro de un aplicativo

Según la figura 6.3, los Persisten Objects son los objetos POJOs que sirven de interfaz entre la aplicación y la base de datos. Esta relación es llevada a cabo mediante un motor (proporcionado por Hibernate) que usa archivos de

configuración tales como el hibernate.properties y archivos XML según la entidad de la base de datos a utilizar.

6.2.4.1. Arquitectura Hibernate

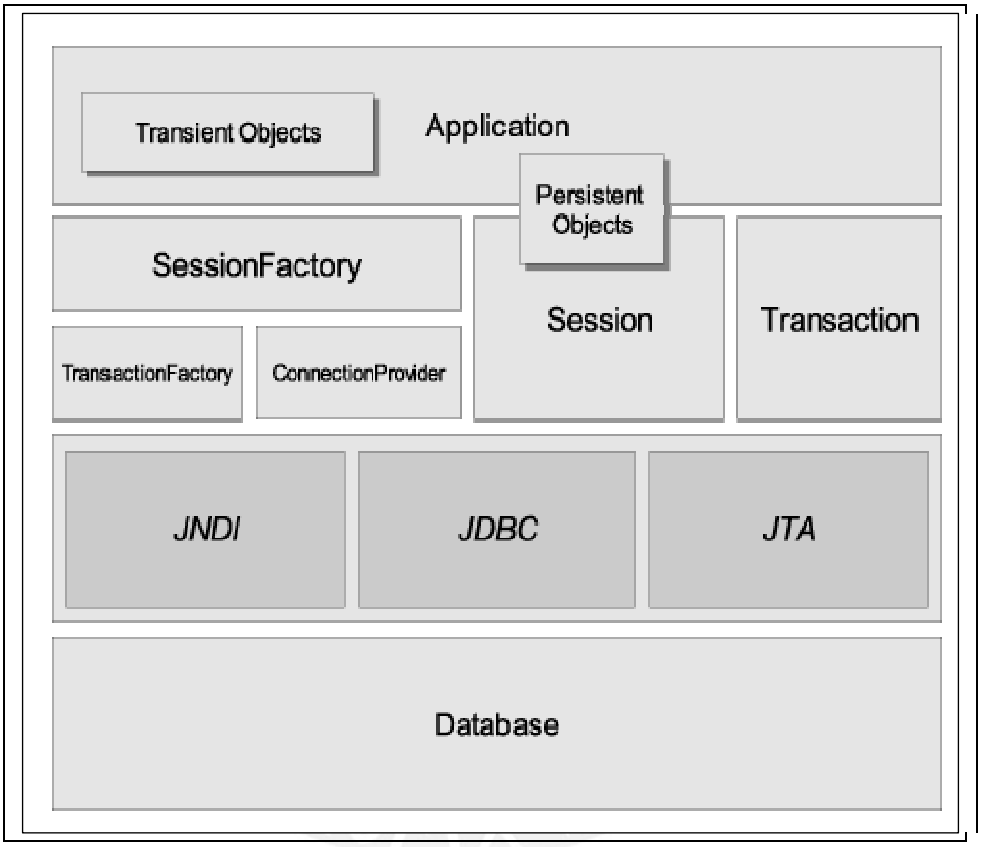

FIGURA 6.13 Arquitectura del Hibernate

Como nos muestra la figura 6.4, la arquitectura del Hibernate nos muestra 4 capas bien diferenciadas: una capa de aplicación, una capa de ejecución del Hibernate (motor), la conexión con la RDBMS y la RDBMS por si misma.

Desde la aplicación, se utilizan los objetos POJOs que guardan los datos generados por la aplicación. De estos POJOs, existen unos que son los que representan entidades del negocio que necesitan

trascender a la aplicación. Estos objetos son los llamados Objetos Persistentes que servirán de transporte entre la aplicación y el RDBMS.

Luego viene la capa propia del motor del Hibernate. La clase más importante que la herramienta nos proporciona, es la clase Session. Esta clase contiene métodos que nos permiten interactuar con la RDBMS como uno para grabar un objeto, posibilidad de creación de consultas, etc.

La siguiente capa es la capa de la conexión a la RDBMS. Se pueden utilizar diversas tecnologías siendo la mas difundida la proporcionada por el mismo lenguaje JAVA, la JDBC (*Java DataBase Connectivity*).

6.2.5. Arquitectura Sistema de Gestión Central

Cabe resaltar que las tecnologías mencionadas anteriormente, necesitan ser adaptadas a las especificaciones que la aplicación requiere. Para esto es necesario diseñar una arquitectura general que permita unir estas tecnologías y que consigan los efectos esperados.

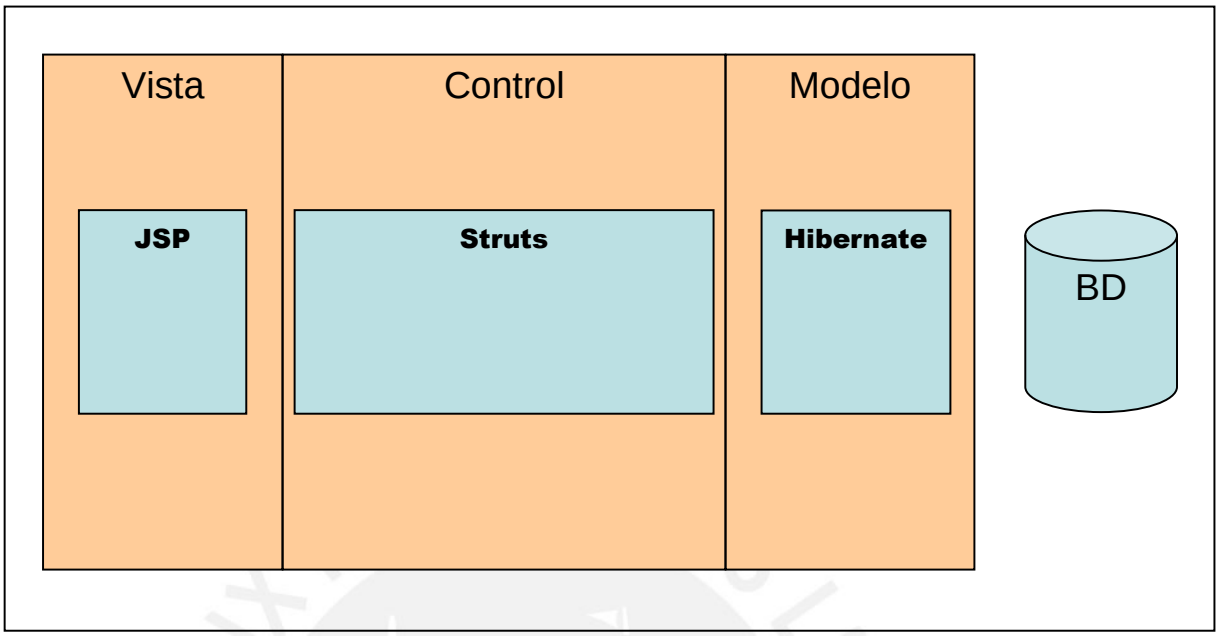

FIGURA 6.14 Arquitectura Sistema de Gestión Central (SGC)

Dado que se eligió como base el patrón MVC (*Model-View-Controller*), es necesario definir clases que nos permitan la interacción entre las capas (como se nota en la figura 6.5). Para una mejor explicación, se organizarán las clases de diseño según la relación entre capas a la que pertenezcan.

6.2.5.1. Relación Vista - Control

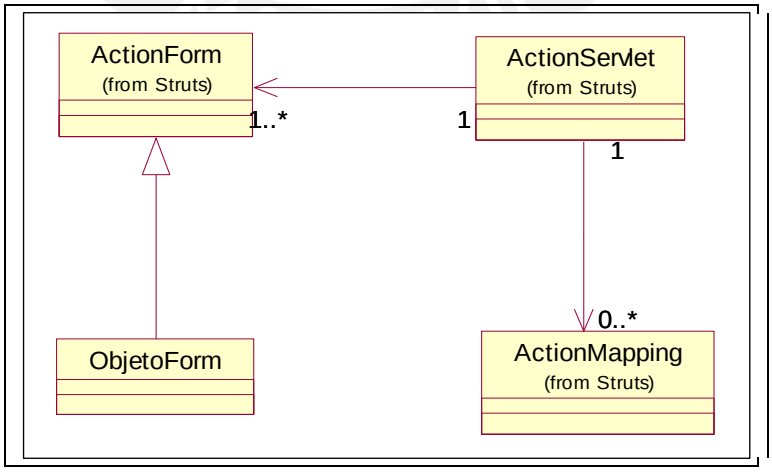

FIGURA 6.15 Diagrama de clases Vista - Control

Se procede a crear clases del tipo ActionForm (ObjetoForm) que serán instanciadas por el ActionServlet que recoge las peticiones HTTP (como se explicó antes). Esta clase ObjetoForm se relaciona directamente con un formulario HTML por lo que es creado siempre que se requiera utilizar datos de este (como se aprecia en la figura 6.6).

6.2.5.2. Relación Control – Modelo

Se procede a crear clases del tipo Action (FuncionAction) según el caso de uso del negocio que se desee implementar. Para organizar de una mejor manera las acciones que realiza el sistema, estas se agrupan según la entidad que las realiza principalmente (no significa que no puedan participar otras entidades del negocio en la realización del caso de uso) definiendo clases (ControlEntidad) y delegándoles la realización de estos (como se aprecia en la figura 6.7).

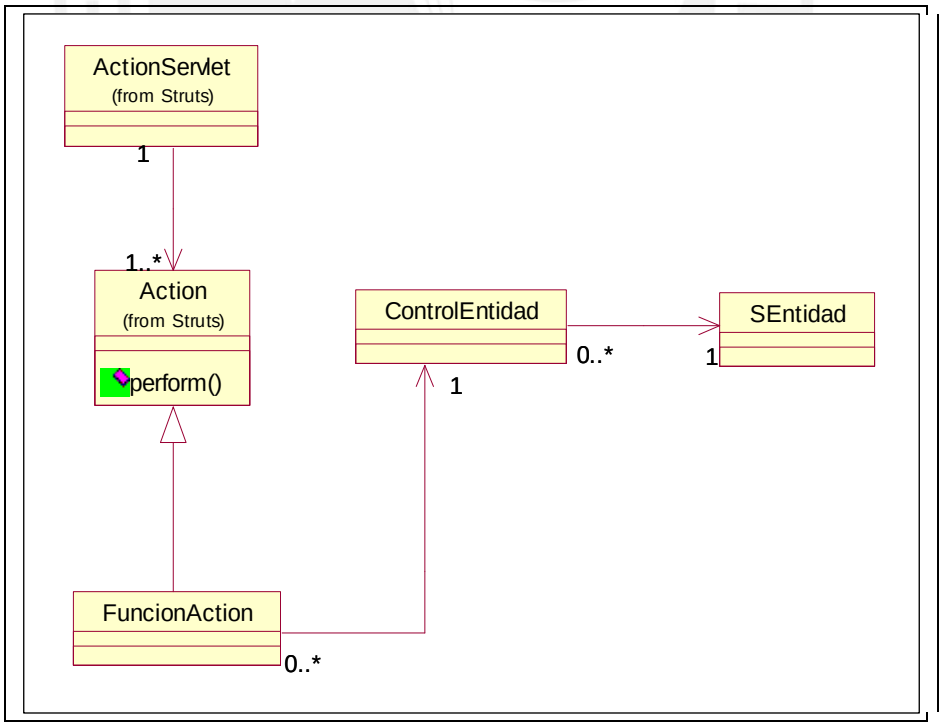

FIGURA 6.16 Diagrama de clases Control - Modelo

Estos objetos (ControlEntidad) se comunican con una clase del tipo SEntidad que es la que implementa la lógica del propia del negocio y se comunica con el Hibernate para poder interactuar con la base de datos.

6.2.5.3. Modelo - Hibernate

Según la entidad de la base de datos que se desee relacionar con una clase de JAVA, se crean las clases AbstractEntidad, Entidad y EntidadDAO.

En la clase AbstractEntidad se definen los atributos del objeto persistente que será correspondido con una tabla de la base de datos.

La clase Entidad hereda de AbstractEntidad y modifica el constructor tomando los parámetros que se requiera según sea el caso.

La clase EntidadDAO es la que se encarga de utilizar los métodos que Hibernate, mediante la clase Session, nos dispone para la interacción con la base de datos además de utilizar los objetos del tipo Entidad (como se aprecia en la figura 6.8).

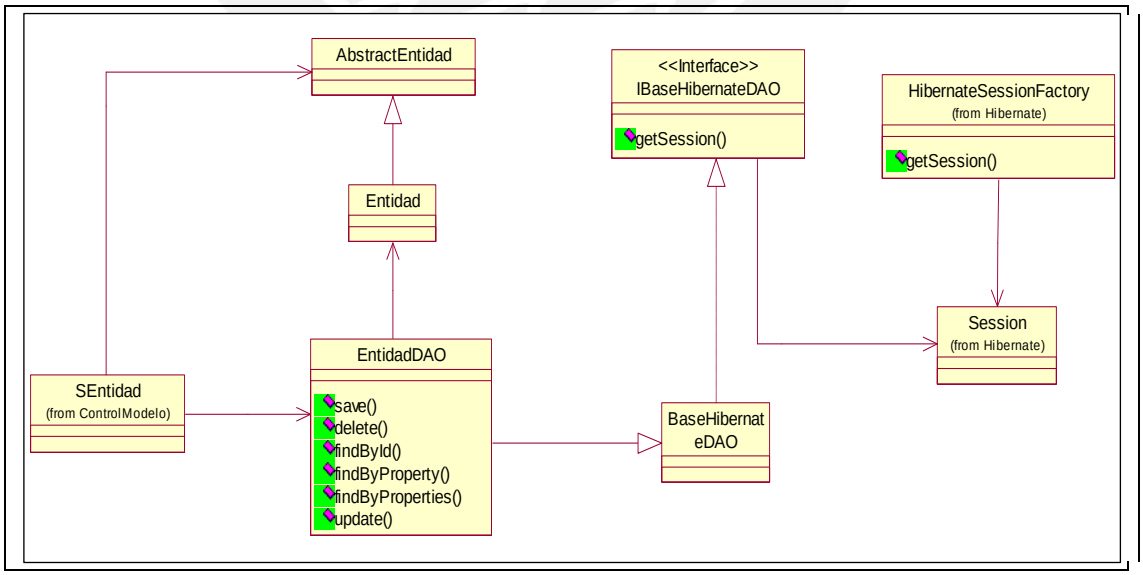

FIGURA 6.17 Diagrama de clases Modelo – Hibernate

6.3. Diagramas de secuencia

Son diagramas que muestran las interacciones entre los objetos organizados en una secuencia temporal mostrando los mensajes intercambiados entre estos. [UML01]

A continuación se muestran los diagramas de secuencias para casos de uso clave dentro del SGC (Sistema de Gestión Central).

6.3.1. Caso de uso Responder pedido de cliente

En las figuras siguientes, se puede ver como se utilizan las clases que definimos para trabajar con el *framework* Struts como son ActionServlet, y las clases Action y ActionForm.

Además, en la figura 6.9 se ve como se listan los pedidos realizados por los clientes mediante el aplicativo móvil. Para esto se utiliza una clase llamada MyGrillaNav que recibiendo ciertos parámetros y llamando a su método construir, se encarga de hacer la consulta a la base de datos y generar una cadena con los resultados (*body*). Posteriormente, la página JSP imprime los resultados en pantalla.

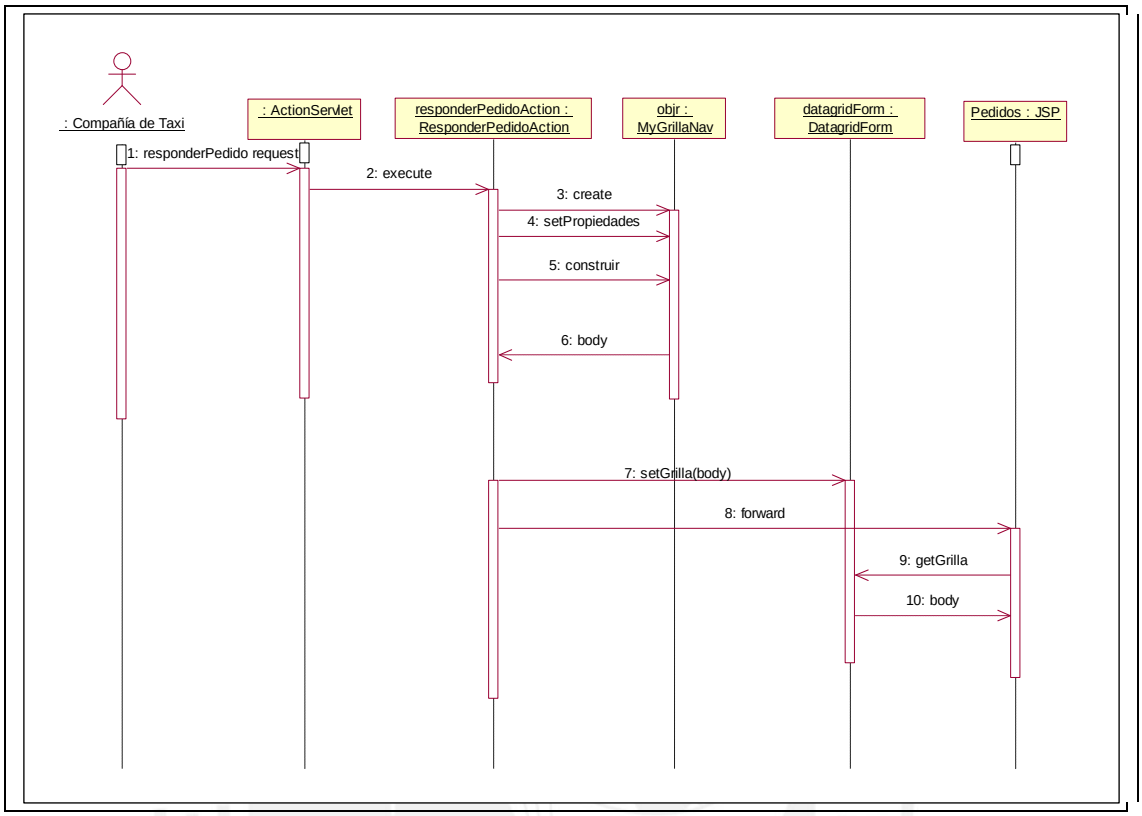

FIGURA 6.18 Diagrama de secuencia caso de uso Responder Pedido Cliente (listar pedidos)

La figura 6.10 muestra la generación de una propuesta por parte de la compañía de taxis. Se utilizan las clases servidoras (nombradas con un prefijo 'S') que son las que sirven de nexo entre la capa de control (en este caso ResponderPedidoPropuestaAction) con la capa de datos.

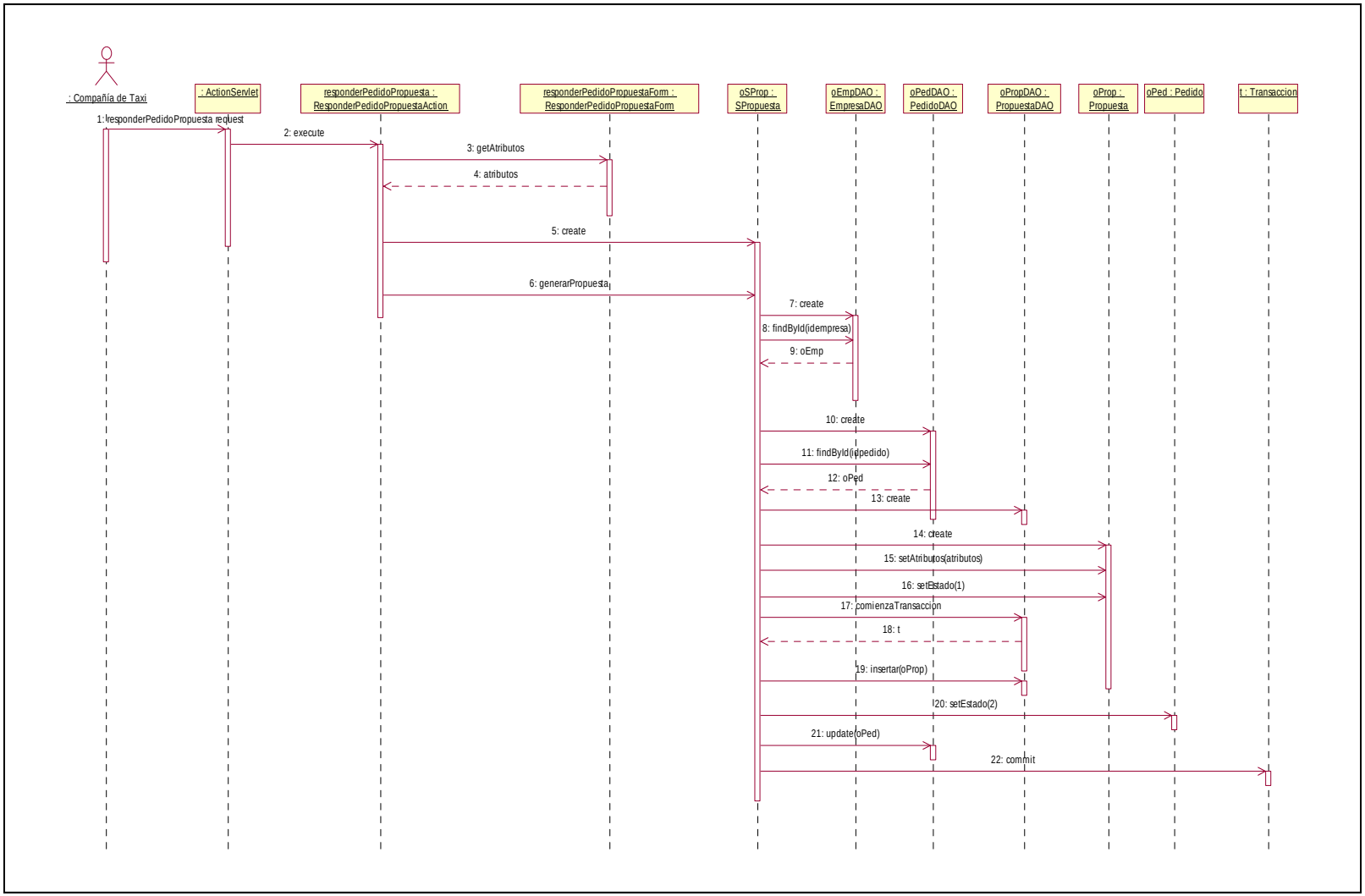

FIGURA 6.19 Diagrama de secuencia caso de uso Responder pedido de cliente (generación propuesta)

6.3.2. Caso de uso Generación de servicio

En la figura 6.11 apreciamos cómo se listan las propuestas contestados por los clientes mediante el aplicativo móvil. Para esto se utiliza una clase llamada MyGrillaNav que recibiendo ciertos parámetros y llamando a su método construir, se encarga de hacer la consulta a la base de datos y generar una cadena con los resultados (body). Posteriormente, la pagina jsp imprime los resultados en pantalla.

En la figura 6.12 se muestra la secuencia encargada de llenar los datos de presentación del formulario de ingreso de un nuevo servicio. En este caso se utilizan las clases servidoras (nombradas con prefijo S) que sirven de intermedio para la comunicación entre la capa de control con la capa del modelo.

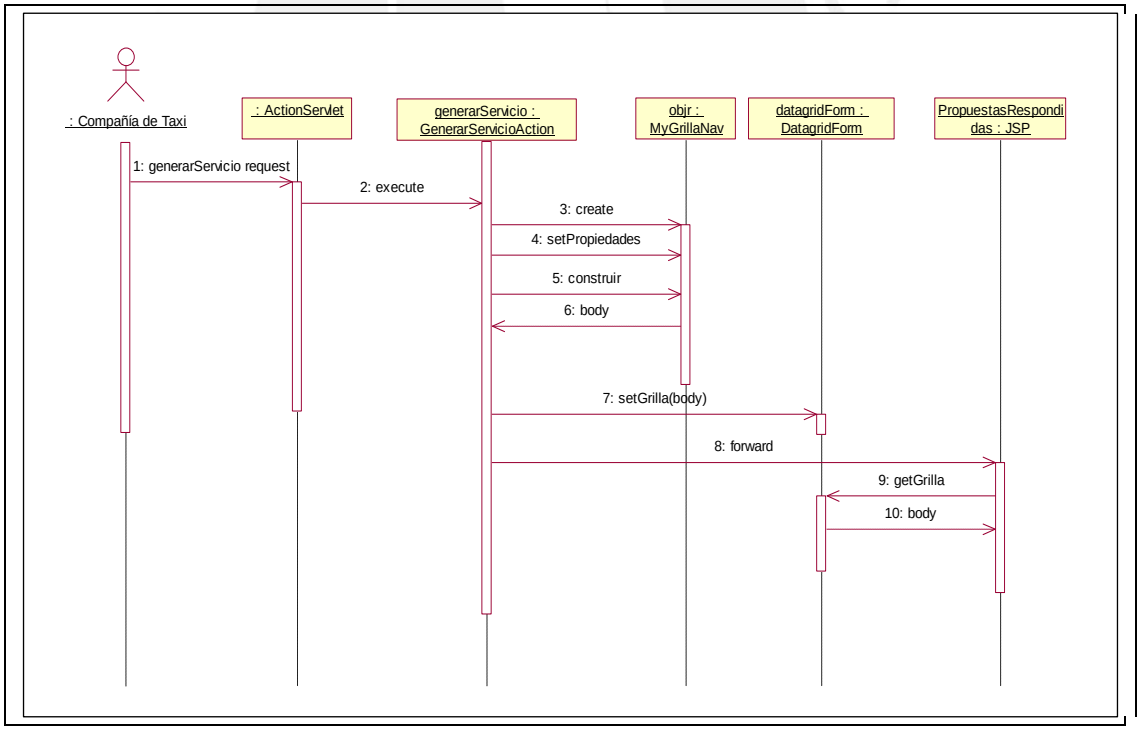

FIGURA 6.20 Diagrama de secuencia caso de uso Generación de servicio (listar propuestas)

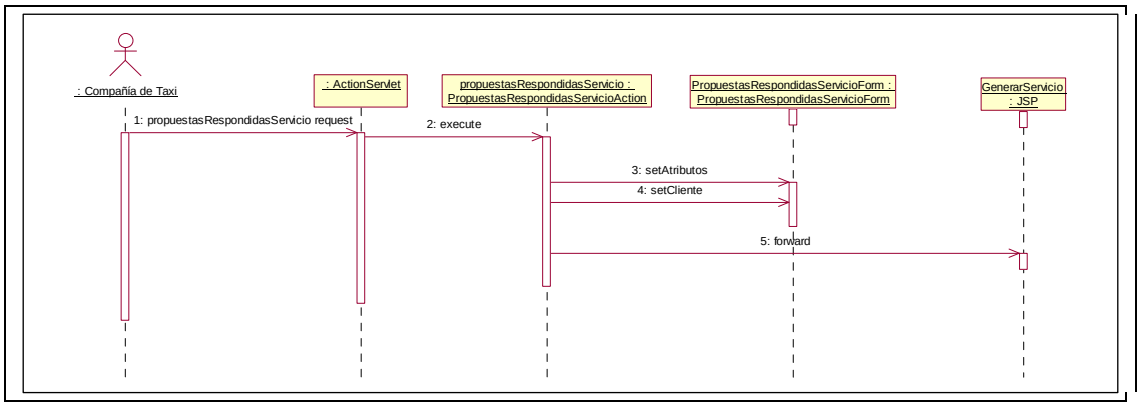

FIGURA 6.21 Diagrama de secuencia caso de uso Generación de servicio (llenar formulario de nuevo servicio)

En figura 6.13 se muestra la secuencia encargada de generar un nuevo servicio. Para este caso también se utilizan las clases servidoras como en los casos anteriores solo que también se utilizan transacción para poder mantener la coherencia de los datos dados posibles errores. Para esto se utiliza la clase Transaction proporcionada por el Hibernate.

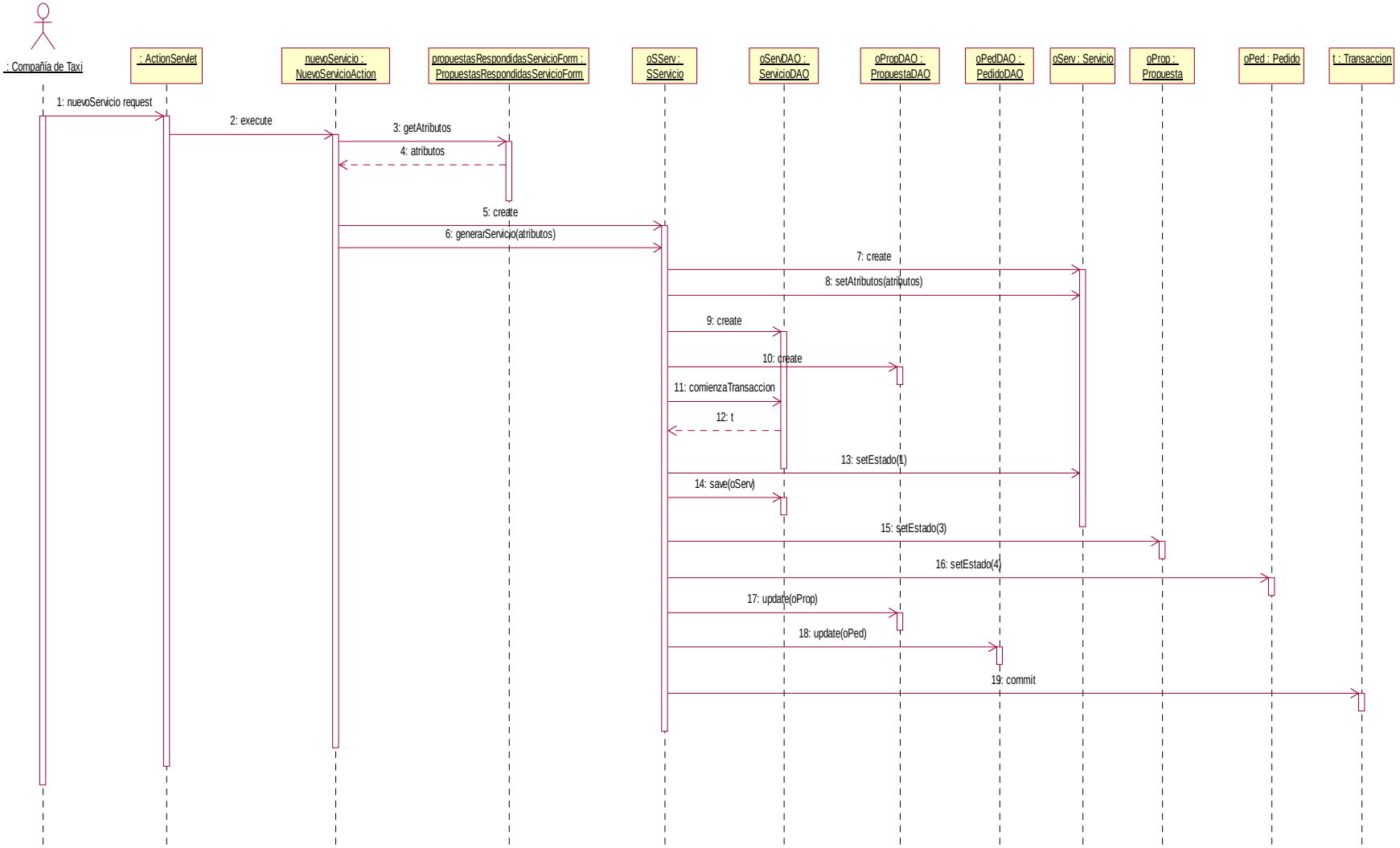

FIGURA 6.22 Diagrama de secuencia caso de uso Generación de servicio (nuevo servicio)

7. Construcción del Software

7.1. Implementación del aplicativo móvil

7.1.1. Herramientas a utilizar

- **Netbeans 4.1 con Netbeans Mobility Pack**

  Herramienta de asistencia para el desarrollo de aplicaciones orientada al lenguaje de programación JAVA. Cuenta con el paquete Netbeans Mobility Pack que nos permite contar con características especiales para el desarrollo de aplicaciones J2ME para dispositivos móviles.

  Características del producto

  - Diseño visual de la interfaz grafica de usuario de la aplicación.
  - Diseño visual del flujo de la aplicación.

- Generación automática de código.

7.2. Implementación del Sistema de Gestión Central

7.2.1. Herramientas a utilizar

- **MyEclipse versión 5.0**

  Herramienta de ayuda para el desarrollo de aplicaciones (IDE) que nos asiste en el ciclo completo de desarrollo de software; codificación, despliegue y pruebas.

  Al ser un proyecto de software libre, siempre se encuentra en permanente actualización, siendo el costo fácilmente accesible para el común de desarrolladores.

  Características del producto

  - Diseño WYSIWYG de formularios Web (JSP y HTML)
  - Editor JSP, XML, AJAX, CSS
  - Diseñador de Struts
  - Soporte para Hibernate

- **PgAdmin III**

  Herramienta de software libre diseñada para los usuarios que administran base de datos del tipo PostgreSQL o versiones comerciales de esta.

Características del producto

- Coloreado de sintaxis al momento de realizar consultas SQL
- Fácil administración de entidades de la base de datos

7.3. Pruebas del software

Un elemento crítico que permite hacer exitoso el proceso de desarrollo de software es la ejecución de los casos de prueba necesarios para asegurar el correcto funcionamiento del producto final. Esto permite a los desarrolladores entregar un producto de alta calidad logrando una mayor satisfacción en los clientes finales del producto.

Las pruebas se definen como la búsqueda sistemática de defectos en los proyectos. Es el proceso de examinar los resultados de procesos comparándolos contra resultados esperados previamente analizados, además de la forma como se tratan las diferencias encontradas. [IBM01]

Dentro del desarrollo de este proyecto, se llegó a la conclusión que solamente se necesitaban 2 tipos de prueba: pruebas de operabilidad y pruebas de integración. Las pruebas unitarias al ser de carácter interno, no se tomaron en cuenta para esta documentación debido a que los resultados esperados son cubiertos con los resultados obtenidos de las pruebas de operabilidad y las de integración.

7.3.1. Pruebas de operabilidad

Las pruebas de operabilidad cubren la totalidad de pantallas del sistema, enfocándose en que la interacción con el usuario sea la correcta pero sin tomar en cuenta los resultados de los procesos realizados.

7.3.2. Pruebas de integración

Las pruebas de integración cubren varias funcionalidades (casos de uso) del sistema, permitiendo analizar un flujo funcional completo incluyendo resultados y tiempos esperados. Dado que un flujo suele ser muy complejo para un análisis simple, este se divide en varios puntos de control que permitirán verificar el correcto funcionamiento de la aplicación por cada punto que vaya transcurriendo.

7.3.2.1. Caso de prueba 1

Precondición. Haberse realizado el caso de uso Autentificación de usuarios (sección 5.2.1.1) teniendo los privilegios necesarios para realizar las funciones del módulo de servicio y del módulo seguridad. Conocer el código (<idempresa>) de la empresa a la que el usuario que esta haciendo uso del sistema pertenece.

Descripción. Se describe un flujo desde la generación de un pedido por parte del cliente desde un dispositivo móvil, pasando por la creación de la propuesta por parte de una compañía de taxi, la aceptación de ésta por parte del cliente, finalizando con la generación de un servicio por parte de la compañía de taxis.

Casos de uso involucrados

- Ingreso de pedidos (sección 5.2.5.1)
- Responder pedido de cliente (sección 5.2.3.3)
- Cancelación de pedido (sección 5.2.5.3)
- Elección de propuestas de pedidos (sección 5.2.5.2)
- Generación de servicio (sección 5.2.3.4)

- Visualización de servicio (sección 5.2.5.4)
- Cancelación de un servicio (sección 5.2.5.6)

Punto de control 1.1

Caso de uso involucrado: Ingreso de pedidos.

En el Sistema de Gestión Central (módulo Servicio) opción 'Pedidos Realizados', la lista de pedidos efectuados deberá mostrar el pedido efectuado corroborando que los datos ingresados en el aplicativo móvil (caso de uso Ingreso de pedido) sea exactamente igual al que aparece en pantalla.

Estos campos son: dirección origen, dirección destino, zona origen, zona destino, fecha y hora. Se debe de anotar el código del pedido (desde ahora referido como impedido) para los posteriores puntos de control.

En el aplicativo móvil, en la opción 'Pedidos Pendientes' deben aparecer los datos relativos a la dirección de origen y destino del pedido efectuado.

Punto de control 1.2a

Caso de uso involucrado: Responder pedido de cliente.

En el Sistema de Gestión Central (módulo Servicio) opción 'Pedidos Realizados', no debe aparecer el pedido efectuado. Para comprobar lo anterior se debe comprobar que el pedido con el código <idpedido> no figure en la pantalla.

El resultado del siguiente *query* (Cuadro 7.1) debe salir 1(para validar la integridad de los datos):

```
SELECT count(*) from pedido pe
INNER JOIN propuesta pr on pe.idpedido=pr.idpedido
```

```
where pe.estado=2 and pr.estado=1 and pe.idpedido=<idpedido>
and pr.idempresa=<idempresa>
```

CUADRO 7.1 *Query* de verificación Punto de control 1.2a

En el aplicativo móvil, en la opción 'Pedidos Aceptados' deberá de aparecer la dirección origen y la dirección destino del pedido realizado con el formato 'De <direccion_origen> a <direccion_destino>'.

Punto de control 1.2b

Caso de uso involucrado: Cancelación de un pedido.

En el aplicativo móvil, en la opción 'Pedidos Pendientes' no deben aparecer los datos relativos a la dirección de origen y destino del pedido efectuado.

El resultado del siguiente *query* (Cuadro 7.2) debe salir 1(para validar la integridad de los datos):

```
SELECT count(*) from pedido pe
INNER JOIN propuesta pr ON pe.idpedido=pr.idpedido
WHERE pe.estado=-1 and pr.estado=-1 and pe.idpedido=<idpedido>
```

CUADRO 7.2 *Query* de verificación Punto de control 1.2b

Punto de control 1.3

Caso de uso involucrado: Elección de propuestas de pedidos.

En el Sistema de Gestión Central (módulo Servicio) opción 'Propuestas Respondidas', debe de aparecer el pedido realizado en el caso de uso Ingreso de pedidos. Para esto se debe de verificar los siguientes datos: cod_ped (código de pedido), dirección origen, dirección destino, fecha (fecha y hora) y tarifa (ingresada en el caso de uso Responder pedido de cliente).

En el aplicativo móvil, en la opción 'Pedidos Aceptados' ya no debe de aparecer el pedido realizado en el caso de uso Ingreso de pedidos.

Para esto se verifica que no haya un pedido con la dirección origen y la dirección destino del pedido realizado (caso de uso Ingreso de pedidos).

El resultado del siguiente *query* (Cuadro 7.3) debe salir 1 (para validar la integridad de los datos):

```
SELECT count(*) from pedido pe
INNER JOIN propuesta pr on pe.idpedido=pr.idpedido
where pe.estado=3 and pr.estado=2 and pe.idpedido=<idpedido>
and pr.idempresa=<idempresa>
```

CUADRO 7.3 *Query* de verificación Punto de control 1.3

Punto de control 1.4a

Casos de uso involucrados: Generación de servicio y Visualización de servicio.

En el Sistema de Gestión Central (módulo Servicio) opción 'Propuestas Respondidas' no debe de aparecer el pedido realizado en el caso de uso Ingreso de pedidos.

En el aplicativo móvil, en la opción 'Pedidos en Progreso' debe de aparecer el pedido realizado en el caso de uso Ingreso de pedidos. Para esto se verifica que no haya un pedido con la dirección origen y la dirección destino del pedido realizado (caso de uso Ingreso de pedidos).

En el aplicativo Web / móvil, en la opción de 'Pedidos en Progreso', seleccionando un pedido, se debe de comprobar que los datos mostrados sean correctos, para esto se debe realizar el siguiente *query* (Cuadro 7.4) a la base de datos:

```
SELECT em.razonsocial as empresa, pr.tarifa as tarifa,
ch.appaterno || ' ' || ch.nombres as chofer,
ma.nombre as marca, us.placa as placa FROM servicio se
INNER JOIN empresa em ON se.idempresa=em.idempresa
INNER JOIN propuesta pr ON se.idpropuesta=pr.idpropuesta
INNER JOIN pedido pe ON se.idpedido=pe.idpedido
INNER JOIN chofer_uservicio cu ON
(se.idunidadservicio=cu.idunidadservicio and cu.fechainicio<=
to_date(se.fechainicio,'DD/MM/YYYY')
and cu.fechafin<=to_date(se.fechainicio,'DD/MM/YYYY'))
INNER JOIN chofer ch ON cu.idchofer=ch.idchofer
INNER JOIN unidadservicio us ON
cu.idunidadservicio=us.idunidadservicio
INNER JOIN marca ma ON us.idmarca=ma.idmarca
WHERE se.estado=1 and pr.estado=3 and pe.estado=4 and
pe.idpedido=<idpedido>
```

CUADRO 7.4 *Query* de verificación Punto de control 1.4a

Punto de control 1.4b

Casos de uso involucrados: Cancelación de servicio

En el móvil, en la opción de 'Pedidos en Progreso', no debe de aparecer el pedido efectuado.

El resultado del siguiente *query* (Cuadro 7.5) debe salir 1(para validar la integridad de los datos):

```
SELECT count(*) from servicio se
INNER JOIN propuesta pr ON se.idpropuesta=pr.idpropuesta
INNER JOIN pedido pe ON se.idpedido=pe.idpedido
WHERE se.estado=-1 and pr.estado=-2 and pe.estado=-2 and
pe.idpedido=<idpedido>
```

CUADRO 7.5 *Query* de verificación Punto de control 1.4b

7.3.2.2. Caso de prueba 2

Precondición: Haberse realizado el caso de uso Autentificación de usuarios (sección 5.2.1.1) teniendo los privilegios necesarios para realizar las funciones del módulo de servicio y del módulo de

configuración y del módulo seguridad. Conocer el código (<idempresa>) de la empresa a la que el usuario que esta haciendo uso del sistema pertenece. Para esto se debe de ingresar al módulo Seguridad, en opción 'Mantenimiento de empresas'.

Descripción. Este caso de prueba comprueba el flujo de una asignación de un chofer a una unidad de servicio. Primero se crea una unidad de servicio y un chofer, para posteriormente hacer la asignación requerida.

Casos de uso involucrados

- Dar de alta a un chofer (sección 5.2.3.1)
- Mantenimiento de unidades de servicio (sección 5.2.2.3)
- Asignación de choferes a unidades de servicio (sección 5.2.3.5)

Punto de control 2.1

Caso de uso involucrado: Dar de alta a un chofer.

Para comprobar que se haya realizado de una manera correcta el caso de uso Dar de alta a un chofer, se debe de apuntar el código del chofer creado (<idchofer>).

En el Sistema de Gestión Central, en el módulo Servicio opción Dar de alta chofer, se debe comprobar que en la lista de choferes, aparezca el chofer creado con los datos ingresados (exista un chofer con código <idchofer>).

Punto de control 2.2

Caso de uso involucrado: Mantenimiento de unidades de servicio.

Para comprobar que se haya realizado de una manera correcta el caso de uso Mantenimiento de unidades de servicio, apuntar el código de la unidad de servicio creada (<idunidadservicio>).

En el Sistema de Gestión Central, en el módulo Configuración opción 'Mantenimiento de unidades de servicio', se debe comprobar que en la lista de unidades de servicio, aparezca la unidad de servicio creada con los datos ingresados (exista un chofer con código <idunidadservicio>).

Punto de control 2.3

Caso de uso involucrado: Asignación de choferes a unidades de servicio.

Se debe de ejecutar el siguiente *query* colocando la fecha del día de la asignación para ver si se encuentra asignado correctamente un chofer a una unidad de servicio para una fecha determinada (además comprueba la relación uno a uno). El resultado del *query* (Cuadro 7.6) debe ser 1.

```
SELECT count(*) FROM chofer_uservicio
WHERE idchofer=1 and idunidadservicio=1 and
fechainicio>=to_date('<fecha_ahora>','DD/MM/YYYY')
and fechafin<= to_date('<fecha_ahora>','DD/MM/YYYY')
```

CUADRO 7.6 *Query* verificación punto de control 2.3

8. Conclusiones, Ampliaciones y Recomendaciones

8.1. Conclusiones

Es posible integrar tecnologías Web con tecnologías móviles para el desarrollo de soluciones de uso general como lo es la propuesta en esta tesis. El uso de tecnologías móviles nos permite utilizar teléfonos celulares como dispositivos que permiten la comunicación entre los clientes y el sistema central. Las tecnologías Web nos facilitan la implementación del SGC (Sistema de Gestión Central) ya que no será necesario la instalación en cada compañía de taxi, sino que estas accederán vía Internet desde cualquier computadora que tenga acceso.

Al utilizar la plataforma JAVA (para el aplicativo Web J2EE y para el aplicativo móvil J2ME) se permite tener una solución robusta que siga estándares definidos para la industria. Esto facilita posteriores mejoras dado que al seguir estándares permite la fácil integración con otras soluciones existentes en el mercado o a desarrollarse.

La utilización de *frameworks* para el desarrollo de software mejora la eficiencia del ciclo de construcción del aplicativo. El utilizar el patrón de diseño MVC (Model View

Controller) permite una fácil integración de frameworks de otras empresas como son Struts (en la parte del Controlador) y Hibernate (en la parte del modelo).

El avance tecnológico en lo referente a dispositivos móviles, hace que cada cierto periodo de tiempo se desarrollen equipos con mayores prestaciones y capacidad de procesamiento que sus antecesores. Este panorama da un futuro alentador al desarrollo de aplicaciones para dispositivos móviles, pero a la vez acarrea el deber de estar en una constante investigación para mantenerse actualizado y no perder la ventaja competitiva.

8.2. Recomendaciones

Para no incurrir en costos excesivos, se recomienda utilizar herramientas de fuente abierta que actualmente se encuentran robustas en el mercado. Estas herramientas son el MyEclipse para el desarrollo Web, PostgreSQL como administrador de base de datos y los siguientes *frameworks* de desarrollo como son el Struts y el Hibernate.

Se recomienda seguir RUP para la gestión de los proyectos para que desde las fases iniciales se considere la orientación a la arquitectura y a los casos de uso con el fin de minimizar la cantidad de errores en la implementación, que podrían reflejarse en retrasos en la ejecución del proyecto.

8.3. Ampliaciones

La aplicación propuesta en este proyecto fue pensada para ser la fuente de datos para los sistemas transaccionales de una empresa de taxis.

Sin embargo la funcionalidad de esta puede ser aumentada automatizando el proceso de gestión de pedidos aprovechando las funcionalidades que nos pueda otorgar un dispositivo móvil.

Por ejemplo, la utilidad de GPS (*Global Positioning System*) nos permitiría conocer la ubicación de las unidades de servicio. Esta funcionalidad integrada a un sistema automático de asignación de recursos, brindaría a las empresas de taxi la posibilidad de poder responder satisfactoriamente mayor cantidad de pedidos de servicio, ya que estos serán asignados automáticamente a una unidad de servicio según la ubicación geográfica del pedido y la ubicación geográfica de esta.

Por ultimo, los datos generados por la aplicación podrán ser almacenados en sistemas *datawarehouse* aplicando la inteligencia de negocios para conocer las particularidades del negocio de taxis, pudiendo inferir por ejemplo la densidad de pedidos por ubicación geográfica, por hora del día, por periodo del año, etc.

GLOSARIO

Definiciones:

Unidad de servicio	Ente encargado de proporcionar el servicio de taxi.
Unidad de taxi	Herramienta utilizada para prestar el servicio de taxi. Usualmente es un automóvil.
Servicio	Acción por la cual se lleva a cabo el transporte de un cliente de un lugar de origen a un lugar de destino. Pedido siendo atendido por una compañía de taxi.
Pedido	Orden de servicio generada por los clientes. Es un posible Servicio.
Propuesta	Plan de servicio diseñado por la compañía de taxi para solucionar la necesidad del Servicio.
Sistemas *Legacy*	Sistemas antiguos.
firmware	Grupo de instrucciones de programa para propósitos específicos, grabados en una memoria del tipo ROM.

Acrónimos:

SGS	Sistema de Gestión Central
J2ME	Java 2 Micro Edition
MVC	*Model-View-Controller*, patrón de diseño de software
URI	*Uniform Resource Identifier* (identificador uniforme de recursos)
POJO	*Plain Old Java Objects*. Este nombre se les da a clases Java

	que no son de algún tipo especial, que no cumplen otro rol más que servir de repositorio de datos.
WYSIWYG	*What You See Is What You Get.* Se refiere al tipo de interfaces gráficas que nos permiten visualizar el resultado final de lo que se encuentra elaborando.
XML	*eXtensible Markup Language*
AJAX	*Asynchronyus Javascript and XML*
CSS	*Cascade Style Sheets*
IDE	*Integrated Development Environment.* Software de ayuda para el desarrollo de aplicaciones.
RDBMS	*Relational DataBase Management Systems.* Sistemas de manejo de base de datos.
CDMA	*Code Division Multiple Access.* Tecnología de comunicación que permite el múltiple acceso de señales dividiendo un único canal de radio según un número seudo aleatorio que se asignó a cada usuario.
TDMA	*Time Division Multiple Access.* Tecnología de comunicación que permite el múltiple acceso de señales dividiendo un único canal según una ranura de tiempo determinado.
GSM	*Global System for Mobile communication.* Estándar de teléfonos móviles europeos.

BIBLIOGRAFIA

[KRU2003] Kruchten P. 'The Rational Unified Process: An introduction'. Addison Wesley.

[DEU1989] L. Peter Deutsch. *Design reuse and frameworks in the Smalltalk-80 system.* En Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Patrones de Diseño. Elementos de software orientado a objetos reutilizable*. Pearson Educación S.A, Madrid 2003. pp 23.

[RED2002] Ueli Wahli, Masaaki Agatsuma, Reginaldo Barbosa, Gert Hekkemberg, Bob McGoogan, Iwan Winoto. *Legacy Modernization with Websphere Studio Enterprise Developer*. IBM Redbooks 2002. pp 78

[GOO01] Google inc. *Google Mobile Maps*. http://www.google.com/gmm/index.html

[GOO02] Google inc. *Froogle*. http://froogle.google.com/

[IEEE01] The Institute of Electrical and Electronics Engineers Inc. *IEEE Std 830-1993 Recommended Practice for Software Requirements Specifications*. IEEE1993

[UML01] James Rumbaugh, Ivar Jacobson, Grady Booch. *The Unified Modeling Language. Reference Manual*. Addison Wesley Longman Inc 1999

[IBM01] IBM Global Services. *Full Lifecycle Testing Concepts. Technique Paper.* International Business Machines Corporation. 2000