UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE QUE PERMITA LA AUTOMATIZACIÓN PARA LA GESTIÓN DE UNA EMPRESA ARRENDADORA DE VEHÍCULOS"

REALIZADO POR:

Arismendi Yaguaraima, Carmelo José

De Sia Coppola, Genaro José

Trabajo de Grado presentado como requisito parcial para optar al Título de:
Ingeniero en Computación

Barcelona, Agosto de 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SOFTWARE QUE PERMITA LA AUTOMATIZACIÓN PARA LA GESTIÓN DE UNA EMPRESA ARRENDADORA DE VEHÍCULOS"

Ing. Aquiles Torrealba

Asesor Académico

Trabajo de Grado presentado como requisito parcial para optar al Título de:
Ingeniero en Computación

Barcelona, Agosto de 2010

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

“DESARROLLO DE UN SOFTWARE QUE PERMITA LA AUTOMATIZACIÓN PARA LA GESTIÓN DE UNA EMPRESA ARRENDADORA DE VEHÍCULOS”

JURADO CALIFICADOR:

Ing. Aquiles Torrealba

Asesor Académico

Ing. Rhonald Rodríguez

Jurado Principal

Ing. Gabriela Veracierta

Jurado Principal

Barcelona, Agosto de 2010

RESOLUCIÓN

De acuerdo al artículo 41 del Reglamento de Trabajos de Grado de la Universidad de Oriente.

"LOS TRABAJOS DE GRADO SON DE EXCLUSIVA PROPIEDAD DE LA UNIVERSIDAD Y SÓLO PODRÁN SER UTILIZADOS A OTROS FINES CON EL CONSENTIMIENTO DEL CONSEJO DE NÚCLEO RESPECTIVO, QUIEN LO PARTICIPARÁ AL CONSEJO UNIVERSITARIO".

DEDICATORIA

Dedico ante todo este trabajo a Dios por darme la fortaleza y la paciencia necesaria para poder culminar mi carrera universitaria. Ese ser universal que nos brinda su amor incondicional y nos da fuerzas para seguir adelante y lograr lo que anhelamos. A él dedico en primer lugar este triunfo.

Por supuesto a mis Padres, Carmelo y Aracelis, quienes han sabido llevarme por buen camino inculcándome los mejores principios y brindándome su amor y apoyo absoluto en todas las etapas de mi vida. Sin duda alguna, todos mis triunfos incluyendo éste son también de ellos, a mis abuelas Josefina, Luisa y a mi Tía Arelis, quienes me han brindado su cariño, comprensión, amor, consejos y apoyo en todas las facetas de mi existencia.

A mi esposa Rossy quien ha sido un ser tan especial en mi vida, mi amor, mi amiga, mi confidente, quien me ha brindado su apoyo en las buenas y en las malas, gracias a ella tengo lo más bello y más grande que me ha dado Dios en esta vida, a mi niña linda Maricelys y a mi niño José Carlos que ya está a pocos días de nacer. Los amo muchísimo, los tres son mi centro de inspiración, mi tesoro, mi adoración.

A mis Hermanos Carlos, Carelis quienes han sido pilares fundamentales para mí en todo momento y en especial a mi querido hermano José Luis, siempre te recordare mi hermanito, fuiste mi guía, como hermano mayor siempre nos llevaste por el buen camino. Te extraño mucho **“JOSE”.**

Carmelo

DEDICATORIA

Ante todo le dedico mi esfuerzo y mis logros a Dios, por darme la fortaleza en los momentos en que mas la he necesitado.

Les dedico este trabajo a mis padres Gennaro De Sia y Alessandra Coppola de De Sia, por su inconmensurable amor y apoyo y por ser las personas más importantes en mi vida.

A mis Hermanas Fabiola y Gennifer, que siempre me han dado palabras de estimulo para culminar esta etapa tan importante, así como a mi cuñado Raffi que también ha sido un gran apoyo moral y un buen ejemplo.

A mis compañeros y amigos con quienes compartí durante la mayoría de esta carrera universitaria: Daniela, Fernando, Jaisfel, Maria, Cesar, David, y tantos otros que aunque no mencione aquí siempre formaran parte importante en mi vida.

Genaro

AGRADECIMIENTOS

Agradecemos al Ing. Aquiles Torrealba por su asesoría, ayuda y consejos dirigidos al desarrollo y culminación de este Trabajo de Grado. Sin duda alguna, uno de los mejores profesores con que cuenta el Departamento de Computación y Sistemas, y además una excelente persona y gran amigo

Agradecemos a nuestros compañeros David, Samuel, Douglas y Cesar por toda su orientación y asistencia durante la elaboración y desarrollo de nuestro trabajo.

Agradecemos a la empresa Oriente-Rent-A-Car por todo el apoyo brindado para la elaboración de este Trabajo de Grado.

Agradecemos también a todos nuestros profesores, los cuales a lo largo de nuestra carrera nos brindaron sus conocimientos y enseñanzas, contribuyendo con nuestra formación académica y personal.

RESUMEN

El presente proyecto consiste en el diseño de una aplicación para automatizar las actividades del proceso de arrendamiento de la empresa Oriente Rent-A-Car C.A, ubicada en el aeropuerto internacional José Antonio Anzoátegui de Barcelona, utilizando la metodología del Proceso Unificado Racional siendo el propósito del mismo, diseñar un sistema automatizado para llevar el control general de los vehículos de la empresa. El proceso de diseño del proyecto, incluyó el levantamiento de información mediante entrevistas realizadas al personal de la empresa, el análisis de la base de datos involucradas, lo cual permitió determinar el comportamiento del sistema actual y así elaborar el modelo del sistema propuesto. Para este se desarrollaron los diagramas de casos de uso, clase de análisis, colaboración, secuencia y finalmente el diagrama de capas de diseño, para la elaboración de las tablas de las base de datos se utilizó Microsoft SQL Server 2005 y en la generación de reportes e interfaz se necesitó la utilización de Microsoft Visual Basic 2005.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

CAPÍTULO 1: PLANTEAMIENTO DEL PROBLEMA

1.1 Introducción

Oriente Rent-A-Car C.A. es una empresa arrendadora de vehículo con muy poco tiempo de servicio en el sector de la movilidad. En Barcelona está a disposición de sus clientes en el aeropuerto internacional José Antonio Anzoátegui, cuenta con una pequeña flota de vehículos de distintos modelos. Su estrategia está en brindar un buen servicio de calidad total, atención personalizada, planes y tarifas que puedan adaptarse a las necesidades de sus clientes.

La satisfacción de sus clientes es su objetivo principal y para ello fomentan su lealtad mediante colaboradores altamente motivados, y la disposición a lograr el máximo rendimiento y ventajosos programas de vinculación de clientes.

En un análisis realizado a los procesos de Oriente Rent-A-Car se pudieron identificar y describir los siguientes aspectos:

Actualmente la gestión de la información de los procesos y procedimientos se realiza de forma manual a través de planillas y formularios, esto genera muchas debilidades y deficiencias para la empresa.

Existe un frecuente retraso en la actualización de datos, debido a la falta de canales de comunicación bien establecidos y la inadecuada estructura organizacional. Un deficiente sistema de administración y control

de datos, ya que varias actividades son realizadas, verificadas y supervisadas por la misma persona.

Se desea diseñar e implementar una base de datos para almacenar y gestionar la información empleada por la empresa dedicada al alquiler de vehículos, teniendo en cuenta los siguientes aspectos:

La empresa dispone de un conjunto de vehículos para su alquiler. Se necesita conocer la matrícula, marca, modelo, color y el precio de alquiler de cada vehículo, el cual está clasificado en clases.

Los datos que interesa recoger de cada cliente son: nombre o razón social, dirección, cedula o Rif, números de teléfono; además, los clientes se diferencian por un código interno de la empresa.

Si un cliente desea solicitar el alquiler de algún vehículo, se pone en contacto con la empresa, suministra sus datos para así proceder con el proceso de arrendamiento respectivo.

Por políticas de la empresa un cliente no puede alquilar más de un vehículo a la vez. De cada reserva interesa registrar la fecha de inicio (cuándo deben entregarse los vehículos al cliente), la fecha de finalización (la fecha prevista de devolución por parte del cliente), el precio total de la reserva y un indicador de si los vehículos han sido devueltos. Además, para cada coche interesa recoger los litros de combustible que contiene y los kilómetros recorridos al momento de su entrega al cliente.

Se desea que la empresa disponga de una adecuada planificación del control y gestión de la información empleada para arrendar vehículos, permitiéndole controlar y evaluar continuamente el desempeño de los

mismos, con el debido soporte informático y con particular atención en el costo de los recursos utilizados. Con ello se espera manejar la información con herramienta de fácil manejo y más amigable al usuario, por ello se debe disponer de un sistema automatizado para el control y gestión de la información con el fin de aumentar su eficiencia y eficacia para llevar registros reales y confiables, trabajando así de una manera más organizada, rápida y efectiva.

Para el desarrollo de esta herramienta tecnológica se aplicarán los lineamientos del Proceso Unificado de Desarrollo de Software, el cual es una metodología para la creación de sistemas y software orientados a objetos; este ofrece un conjunto de modelos, físicos y lógicos que son imprescindibles para la elaboración de cada una de las etapas o fases a desarrollar en el sistema. Esta metodología está enmarcada en tres aspectos relevantes como son:

- Dirigidos por caso de uso.
- Centrado en la arquitectura.
- Es iterativo e incremental.

De allí la importancia de emplear este proceso, por su gran flexibilidad en el momento de elaborar un sistema, ya que ayuda a relacionar las necesidades del usuario con todas las personas implicadas en el proyecto, brinda una clara perspectiva del sistema propuesto, dividiendo el trabajo en mini proyectos los cuales al cumplirse representan un incremento en el desarrollo del mismo; tomando en cuenta que para obtener un software de calidad, se necesita una forma coordinada de trabajar, que gestione el proceso para garantizar su excelencia y esto se logra aplicando las fases del

Proceso Unificado, como son las de inicio, elaboración, construcción y transición.

Estas a su vez se llevan a cabo a través de una o varias iteraciones las cuales constan de cinco flujos de trabajo fundamentales como son: Captura de requisitos, análisis, diseño, implementación y prueba.

La escogencia de este lenguaje se debe a que toda empresa bien constituida debe cumplir con marcos jurídicos para evitar sanciones y multas por parte de la ley, por ello se utilizaron las versión express de visual basic 2005 y SQL server 2005, las cuales son gratuitas.

Para crear y diseñar de manera eficiente la base de datos se trabajará con Microsoft SQL Server 2005, utilizando técnicas que permitan la integridad de los datos, evitando de esta manera cualquier tipo de anomalías de los mismos.

La elaboración de este Proyecto de Investigación será muy importante, ya que se logrará automatizar el sistema de una empresa arrendadora de vehículo, Oriente Rent-A-Car C.A. Y además por ser la primera vez que se desarrollará un sistema como tal, en el departamento de Computación Y Sistema de la Universidad de Oriente, Núcleo de Anzoátegui, como proyecto de investigación.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un software que permita automatizar la gestión de una empresa arrendadora de vehículos.

1.2.2 Objetivos Específicos

- Analizar la situación actual de la gestión de alquiler de vehículos de la empresa.
- Determinar los requerimientos de información, consulta y reporte del nuevo software.
- Modelar la estructura del software y de la base de datos del nuevo sistema.
- Diseñar la interfaz del software que se adapte a los requerimientos de los usuarios.
- Construir los diferentes módulos del software.
- Efectuar las pruebas finales al sistema.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Antecedentes

Para la elaboración de una base teórica amplia y completa necesaria para el desarrollo de este trabajo de grado, se tomaron en cuenta algunos proyectos realizados anteriormente en la Universidad de Oriente que aplicaron en su desarrollo la metodología utilizada en este trabajo. Estos proyectos se nombran a continuación:

- **Sifontes, C. (2005)** en su trabajo "Diseño de un sistema de información automatizado para el alquiler de maquinarias de la empresa FERRETECA", realizó un sistema de información para la empresa antes mencionada, que permitió automatizar el proceso de alquiler de maquinarias para la construcción. El sistema de información diseñado logró controlar el proceso evitando errores humanos. **[1]**

- **Malpica, M. (1.998)** en su trabajo "Automatización del proceso de acceso a la entrada de maquinarias y vehículos de la empresa Costa Norte Construcciones", diseñó un sistema de información que permitió controlar tanto el acceso de maquinarias como de vehículos a la empresa. Se adicionaron otros controles como brazo automático, lectura de carnet, todo ello para automatizar el proceso completo. Con el desarrollo de esta investigación se logró disminuir el tiempo y reducir ciertos costos. **[2]**

- **Herrera, A. (1998)** realizó una tesis sobre el “Diseño de un sistema de control de chequeo y estatus de flota de vehículos adscritos al SETRA”. En este trabajo se diseñó un sistema de información para el control de vehículos. Además se diseño un prototipo o panel de control GPS para mostrar la ubicación de los mismos en las diferentes rutas. **[3]**

- **Quintero, M. (2001)** realizó una investigación titulada "Implantación de un Nuevo Sistema de Control de Inventarios e Investigación Acerca de los Resultados de su Aplicación en la Impsat S.A. de la Universidad Metropolitana", planteándose como objetivo general, estudiar y realizar el Sistema de Control de Investigación de la empresa y aplicar los ajustes, correcciones y adaptaciones que sean necesarias para garantizar el abastecimiento. **[4]**

- **Velásquez, O. (2003)** en su trabajo “Desarrollo de un sistema automatizado para el control de actividades asociadas al área de resguardo para una instalación aduanera en Guanta, Estado Anzoátegui”, en este trabajo se realizo un sistema para automatizar algunos de los procedimientos asociados al normal funcionamiento de La Aduana Principal de Guanta, específicamente el control de las actividades asociadas al área de resguardo.**[5]**

2.2 Fundamentos Metodológicos

2.2.1 El Proceso Unificado de Desarrollo de Software

El Proceso Unificado, como su nombre lo indica, es un proceso que unifica el desarrollo de software mediante un conjunto de actividades necesarias para transformar los requisitos de un usuario o de un grupo de éstos en un sistema de software. **[6]**

Su aplicación se fundamenta en una herramienta esencial para tal fin, el Lenguaje Unificado de Modelado (UML), sumamente versátil y poderosa para la elaboración y diseño de los diferentes planos, trazas o perspectivas del sistema software.

El Proceso Unificado utiliza el UML para expresar gráficamente todos los esquemas de un sistema software. Pero, realmente, los aspectos de este proceso unificado son tres:

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e incremental.

2.2.1.1 El Proceso Unificado dirigido por casos de uso

Un sistema de software está creado o desarrollado para brindar algún servicio útil para sus usuarios. Estos son agentes externos que se interrelacionan con el sistema creado o desarrollado. Luego proveen de

Cierta información pertinente al sistema, consecuentemente éste ejecuta una serie de acciones para proporcionar algún resultado particular importante.

Este intercambio de acciones se conoce como caso de uso, así entonces, se define caso de uso como un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso se emplean para especificar los requisitos funcionales de un sistema, además que guían su diseño, implementación y prueba, en suma, guían ciertamente el proceso de desarrollo.

Dirigido por casos de uso significa que el proceso de desarrollo avanza a través de una serie de flujos de trabajo originados precisamente en los casos de uso. Es un importante mecanismo base para establecer la construcción de los diferentes planos o perspectivas de un sistema. Los casos de uso no se desarrollan independientemente, es decir influyen en la arquitectura del sistema a la vez que son influidos por ésta, en consecuencia la arquitectura del sistema como los casos de uso maduran según avanza el ciclo de desarrollo.

2.2.1.2 El Proceso Unificado centrado en la arquitectura

La arquitectura del software plantea diferentes planos del mismo software con características propias, esta idea es análoga a la arquitectura de un edificio, el cual tiene diferentes tipos de planos: eléctrico, civil y mecánico entre otros, es decir las diferentes visiones o perspectivas del mismo inmueble. De igual forma sucede con el software.

La descripción de esta arquitectura se establece mediante un conjunto de vistas del sistema o vistas de los modelos del sistema (vista del modelo de casos de uso, vista del modelo de análisis, vista del modelo de diseño, vista del modelo de despliegue y vista del modelo de implementación). La descripción de la arquitectura describe las partes del sistema, importantes que comprendan todos los interesados en él.

La arquitectura persigue ciertos objetivos fundamentales como la comprensibilidad, la capacidad de adaptación al cambio y la reutilización. Pero no se puede hacer referencia a estos objetivos sin apuntar a la relación de vínculo de los casos de uso con la arquitectura. Estos dos planteamientos deben equilibrarse ya que ninguno de ellos por separado garantiza obtener un producto con éxito. Debe existir una interacción entre los casos de uso y la arquitectura, unos representan la función y otra la forma.

La arquitectura debe diseñarse para permitir la evolución del sistema, tanto en su desarrollo inicial como a lo largo del tiempo.

2.2.1.3 El Proceso Unificado iterativo e incremental

Para abordar el desarrollo de software se aconseja dividir el trabajo en partes más pequeñas o mini proyectos conocidos como iteraciones los cuales satisfactoriamente ejecutados derivan incrementos. Las iteraciones se refieren a los pasos en el flujo de trabajo (requisitos, análisis, diseño, implementación y prueba) y los incrementos, al crecimiento del sistema o producto. Para un mejor desempeño las iteraciones deben seleccionarse y ejecutarse de una forma planificada, es decir deben estar controladas, ver (Figura 2.1).

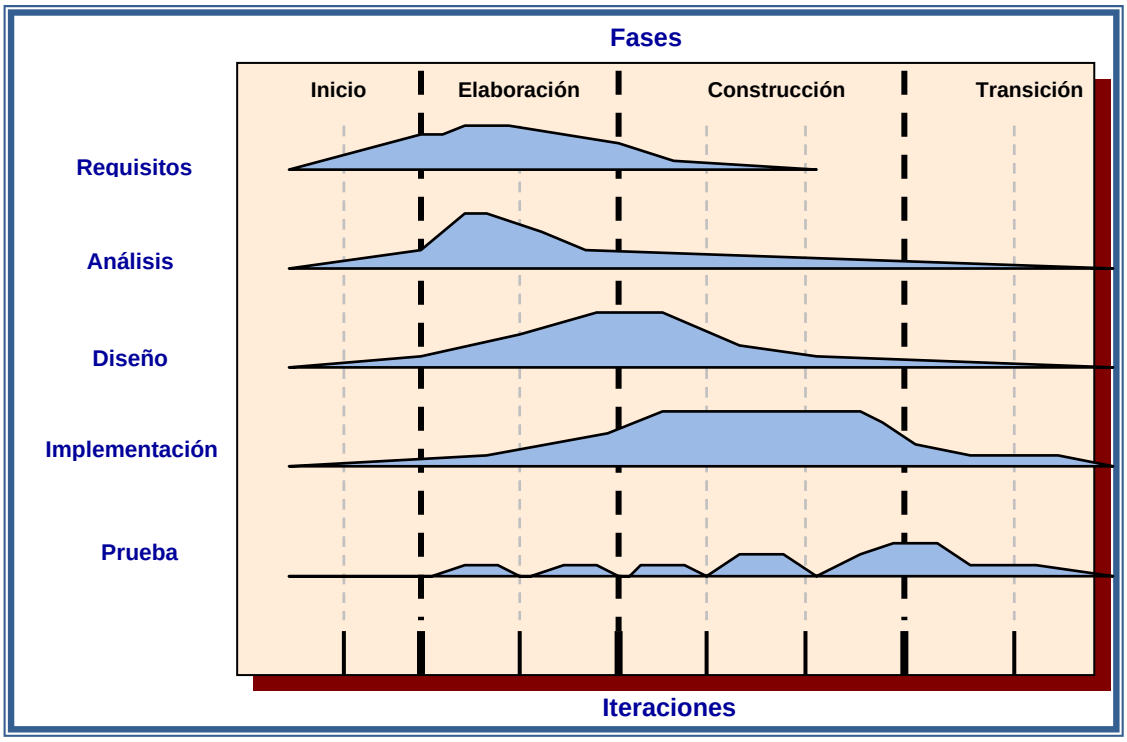


Figura 2.1 Proceso Iterativo e Incremental

Una iteración se establece en dos factores: la iteración trata un grupo de casos de uso los cuales amplían la utilidad del producto hasta ahora desarrollado. En el otro factor, la iteración trata los riesgos más importantes. Las iteraciones siguientes se fundamentan sobre lo que quedó desarrollado al final de la última iteración. Las iteraciones como mini proyectos se inician con los casos de uso, es decir la captura de requisitos y continúan a través del trabajo de desarrollo siguiente (análisis, diseño, implementación y prueba), es decir una iteración es la ejecución de un flujo de trabajo, esto conlleva a la creación del código ejecutable gracias a los casos de uso desarrollados en la iteración.

En cada iteración se identifican y se especifican los casos de uso emblemáticos, se crea un diseño utilizando la arquitectura seleccionada

como guía, se implementa el diseño a través de componentes y se verifica si éstos satisfacen los casos de uso. Si una iteración satisface sus objetivos, se continúa con la siguiente fase, en caso contrario se debe revisar las decisiones previas y probar un nuevo enfoque.

2.2.2 Fases del Proceso Unificado de Desarrollo de Software

Cada ciclo de la vida del software se subdivide en cuatro fases (inicio, elaboración, construcción y transición). Cada fase se descompone en iteraciones con sus respectivos incrementos y concluyen con la disponibilidad de un conjunto de modelos o documentos desarrollados hasta conseguir un estado predefinido, llamado hito, el cual permite la toma de decisiones importantes para así continuar la fase siguiente, ayudando a controlar el progreso del trabajo por cada fase. A través del seguimiento del tiempo y esfuerzo aplicado en cada fase se consigue un conjunto de datos útiles para la estimación de tiempo y recursos para otros proyectos, además de su asignación tomando en cuenta el tiempo de duración. **[6]**

2.2.2.1 Fase de Inicio

El objetivo general de esta fase es poner en marcha el proyecto de desarrollo, donde se desarrolla previamente el análisis del negocio para delimitar el alcance (ámbito) del sistema, el cual es necesario para comprender la arquitectura y así ubicar y enfrentar tempranamente los riesgos dentro de los límites del sistema.

Se debe establecer una arquitectura capaz de soportar el ámbito del sistema, conocida por “arquitectura candidata”. En esta fase se determinan las principales funciones del sistema mediante un modelo de casos de uso

comprendido por los casos de uso más críticos, además de identificar y dar prioridad a los riesgos más importantes, estas últimas son las actividades más relevantes de la fase de inicio.

2.2.2.2 Fase de Elaboración

Para esta fase se recopila el mayor número de requisitos funcionales, aún pendientes, mediante los casos de uso y se diseña la arquitectura del sistema. Dicha arquitectura se expresa a través de las vistas de todos los modelos del sistema, es decir, vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y modelo de despliegue. La resultante de esta fase es la línea base de la arquitectura. Con esta fase se tiene la ventaja de poder planificar las actividades y estimar los recursos necesarios para ultimar el proyecto. Consecuentemente se aclaran dudas acerca de la suficiente estabilidad de los casos de uso, de la arquitectura y el plan; además de discernir sobre si están completamente controlados los riesgos como para seguir en el esfuerzo de desarrollo.

2.2.2.3 Fase de Construcción

La idea fundamental de esta fase es culminar y entregar un producto software en su versión operativa inicial conocida como “versión beta”. El producto debería poseer la calidad adecuada y la seguridad de cumplir los requisitos para así emprender su aplicación en la comunidad de usuarios. Para alcanzar todo lo anterior se deben detallar todos los casos de uso y escenarios restantes, modificar si es pertinente la descripción de la arquitectura, dejar cerrados los modelos de análisis, diseño e implementación continuados en los flujos de trabajo de las iteraciones

adicionales. Además integrar los subsistemas y someterlos a prueba y hacer lo propio con todo el sistema.

Finalmente en esta fase el producto contiene todos los casos de uso que la dirección y el cliente han convenido en desarrollar para la versión beta. Pero es aquí donde se establece y aclara la expectativa acerca de sí cubre suficientemente el producto desarrollado, las necesidades de algunos usuarios para hacer realidad la entrega de la versión inicial.

2.2.2.4 Fase de Transición

Esta fase tiene como objetivos fundamentales, cumplir con todos los requisitos establecidos en fases anteriores, hasta satisfacer a los usuarios y preparar los aspectos relativos a la operación del sistema en el contexto del usuario, implicando la corrección de los posibles defectos señalados por éstos en la versión beta.

La fase de transición abarca actividades como la fabricación, formación de cliente, entrega de una línea de ayuda y asistencia, y la corrección de defectos encontrados posterior a la entrega del software.

2.2.3 Flujos de Trabajo del Proceso Unificado

El Proceso Unificado de Desarrollo de Software posee la virtud de estructurar el esfuerzo de desarrollo de una manera sencilla, su cronología operativa se hace entender fácilmente, es decir el hecho de establecer un sistema de software en ciclos de vida y que cada uno de éstos a su vez se constituya en fases (inicio, elaboración, construcción y transición) consecuentemente divididas, estas últimas, en iteraciones y que estas iteraciones se concreten

en los cinco flujos de trabajo (requisitos, análisis, diseño, implementación y prueba), es sencillamente metódico y lógico. **[7]**

2.2.3.1 Captura de Requisitos

Para la ejecución de este flujo fundamental se debe iniciar con la creación de un modelo representativo del sistema a desarrollar, este modelo contiene los requisitos funcionales (casos de uso), entonces se habla de modelo de casos de uso.

A grandes rasgos en este flujo se identifican, documentan y detallan los casos de uso representativos y se identifican igualmente los actores. Esto conlleva a establecer el contexto del sistema. Se establece un modelo de casos de uso contenido por los requisitos funcionales y no funcionales de casos de uso concretos, elaborado mediante una descripción general, un conjunto de diagramas y una descripción detallada de cada caso de uso. Los casos de uso son una herramienta casi natural y sistemática para la captura de requisitos funcionales poniendo especial interés al valor que tendrá para cada usuario individual.

2.2.3.2 Análisis

La palabra análisis denota por sí misma la esencia del flujo de trabajo homónimo, es decir, se busca obtener una comprensión y descripción más precisa de los requisitos, para ser más fácil de mantener y estructurar el sistema y su arquitectura. En el análisis se puede estudiar y descifrar más sobre aspectos internos del sistema, como también elaborar una estructura de los requisitos para facilitar su comprensión, su preparación, su modificación y mantenimiento. Esta estructura se funda en las clases de

análisis y paquetes las cuales son independientes de las basadas en casos de uso obtenidas en el flujo de requisitos.

Se debe señalar que el modelo de análisis hace abstracciones y evita resolver algunos problemas y tratar algunos requisitos, por tanto se posponen al diseño y a la implementación.

2.2.3.3 Diseño

En este flujo moldeamos el sistema y establecemos su forma (arquitectura) con la intención de dar soporte a sus requisitos funcionales y no funcionales. El diseño se vale del modelo de análisis obtenido en el flujo de igual nombre, porque proporciona una comprensión detallada de los requisitos, siendo esencial para los propósitos establecidos por el diseño.

Ahora bien, estos propósitos abarcan la comprensión profunda de temas relacionados con los requisitos no funcionales, además de las limitaciones relacionadas a los lenguajes de programación, sistemas operativos, tecnologías de gestión de transacciones, tecnologías de interfaz de usuario, entre otros.

El diseño también se plantea la misión de establecer adecuadas entradas y correctos puntos de partida para el flujo de implementación mediante la captura de requisitos o subsistemas individuales, interfaces y clases.

2.2.3.4 Implementación

En este flujo de trabajo se implementa el sistema en componentes como código fuente, y ejecutables entre otros. El objetivo principal es desarrollar la arquitectura y sistema como un todo.

Previamente, se deben planificar las integraciones de sistema necesarias en cada iteración, a través de un enfoque incremental, es decir una serie de pasos pequeños y manejables. Seguidamente se debe distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue. Luego se prueban los componentes individualmente, posteriormente se integran compilándolos y enlazándolos en ejecutables y comprobando todo el sistema.

2.2.3.5 Prueba

Aquí se verifica el resultado de la implementación probando las construcciones internas así como las versiones finales del sistema a entregarse.

En esta fase se comprueba la correcta operación del sistema por medio de una serie de pruebas y en caso de producirse alguna falla, volver a la fase anterior, corregir los errores y continuar con las pruebas hasta asegurar su correcto funcionamiento.

2.2.4 El Lenguaje Unificado De Modelado (UML)

En todas las disciplinas de la Ingeniería se hace evidente la importancia de los modelos; por describir el aspecto y la conducta de "algo" existente en un

estado de desarrollo, como también en un estado de planeación. Es aquí cuando los diseñadores del modelo deben investigar los requerimientos del producto terminado y dichos requerimientos pueden incluir áreas tales como: funcionalidad y confiabilidad. Además, el modelo es dividido en un número de vistas, las cuales describen un aspecto específico del producto o sistema en construcción.

El modelado sirve no solamente para los grandes sistemas, aún en aplicaciones de pequeño tamaño se obtienen beneficios de modelado; sin embargo, es un hecho que entre más grande y más complejo es el sistema, más importante es el papel del modelado por una simple razón: "El hombre hace modelos de sistemas complejos porque no puede entenderlos en su totalidad".

Estos modelados son realizados por medio de un lenguaje llamado UML, independientes de los métodos de análisis y diseño. Existen diferencias importantes entre un método y un lenguaje de modelado. Un método es una manera explícita de estructurar el pensamiento y las acciones de cada individuo, le dice al usuario qué hacer, cómo hacerlo, cuándo hacerlo y por qué hacerlo; mientras que el lenguaje de modelado carece de estas instrucciones. Los métodos contienen modelos y esos modelos son utilizados para describir algo y comunicar los resultados del uso del método, su emblema lo podemos visualizar en la (Figura 2.2).

Figura 2.2 Logo de UML

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar:** UML permite expresar de una forma gráfica un sistema de forma que otro lo pueda entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

2.2.4.1 Beneficios de UML

- Mejor tiempo total de desarrollo (de 50 % o más).
- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado para ser utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos. **[8]**

2.2.4.2 Diagramas UML

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones ofreciendo una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas incluyendo los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de secuencia.
- Diagrama de objetos.
- Diagrama de colaboración.
- Diagrama de estados.
- Diagrama de actividades.
- Diagrama de componentes.
- Diagrama de despliegue.

2.2.4.3 Modelo conceptual de UML

Un elemento esencial en el modelo conceptual de UML son los bloques de construcción, Estos junto con las reglas que se aplican sobre esos bloques y los mecanismos comunes de UML definen la totalidad del modelo conceptual.

Bloques de construcción

Existen tres tipos de bloques de construcción:

- **Elementos:** Son los modelos UML (clases, casos de uso, estados, anotaciones, entre otros).
- **Relaciones:** Ligan elementos entre sí, establecen la forma en que interactúan.
- **Diagramas:** Representación gráfica de un grupo de elementos y sus relaciones.

2.2.4.3.1 Elementos

Los elementos más importantes que conforman este lenguaje son:

- **Casos de uso:** Un caso de uso es una descripción de un conjunto de acciones ejecutadas por el sistema tras la orden de un agente (llamado actor) que puede ser el usuario de la aplicación, la propia aplicación, otro caso de uso o un elemento externo (hardware). Los casos de uso suelen representar funcionalidades del sistema; se representan como una elipse en cuyo interior figura el nombre (lo más descriptivo posible) del caso de uso (ver Figura 2.3).

Figura 2.3 Representación de un caso de uso

- **Clases:** En una clase se agrupan todos los objetos que comparten los mismos atributos, métodos y relaciones. Los atributos son características y propiedades comunes en todos los objetos de la clase. Los métodos son operaciones que deben cumplir las instancias de la clase. Las clases se representan como un

rectángulo donde figuran el nombre de la clase, sus atributos y sus métodos (ver Figura 2.4).

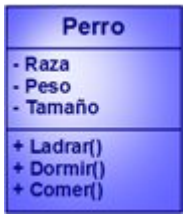


Figura 2.4 Representación de una clase

2.2.4.3.2 Relaciones

- **Dependencia:** Una dependencia es una relación de uso entre dos elementos (un elemento utiliza a otro). Una relación de dependencia entre dos elementos implica que los cambios que se produzcan en un elemento pueden afectar al otro pero no necesariamente a la inversa. Las dependencias se representan con una línea dirigida discontinua (ver Figura 2.5).

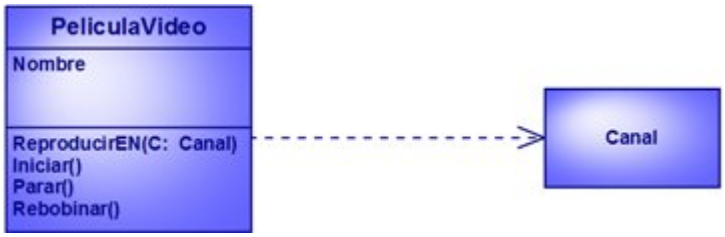


Figura 2.5 Representación de la Relación de dependencia entre dos clases

- **Asociación:** Una asociación es una relación estructural entre varios elementos. Una relación de asociación implica que los objetos de los distintos elementos de la relación están conectados entre sí y se pueden comunicar. Una relación de asociación se

representa gráficamente con una línea continua entre los elementos relacionados (ver Figura 2.6).

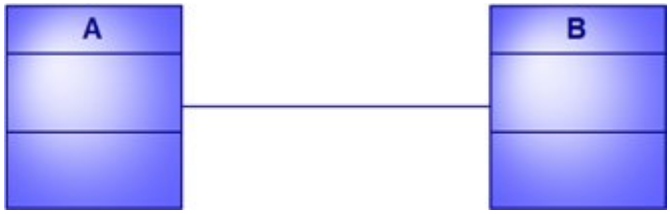


Figura 2.6 Representación de una relación de asociación

- **Generalización:** Una generalización es una relación de especialización. Los elementos especializados (hijos) son elementos que se derivan de un elemento general (padre). Los elementos hijos mantienen la estructura y el funcionamiento del elemento padre pero de una forma más especializada. Su representación gráfica es la de una línea dirigida con punta triangular (ver Figura 2.7).

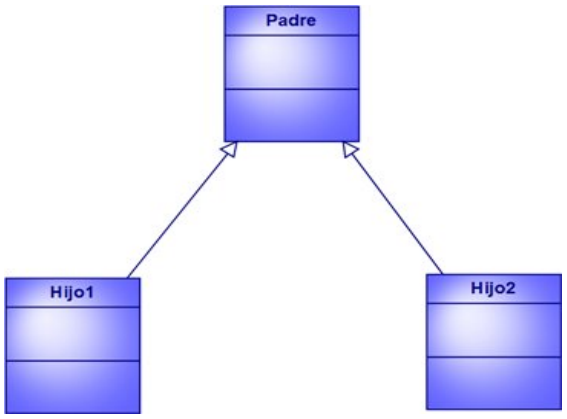


Figura 2.7 Relación de generalización

2.2.4.3.3 Diagramas

- **Diagramas de casos de uso:** Un diagrama de casos de uso es un diagrama que muestra un conjunto de casos de uso con sus relaciones y los actores implicados. Es un diagrama que sirve para modelar la vista estática de un programa. La vista estática nos permite visualizar el comportamiento externo del programa; de esta forma, conseguimos conocer qué es lo que debe hacer el programa independientemente y de cómo lo haga y sabremos los elementos que interactúan con el sistema (ver Figura 2.8).

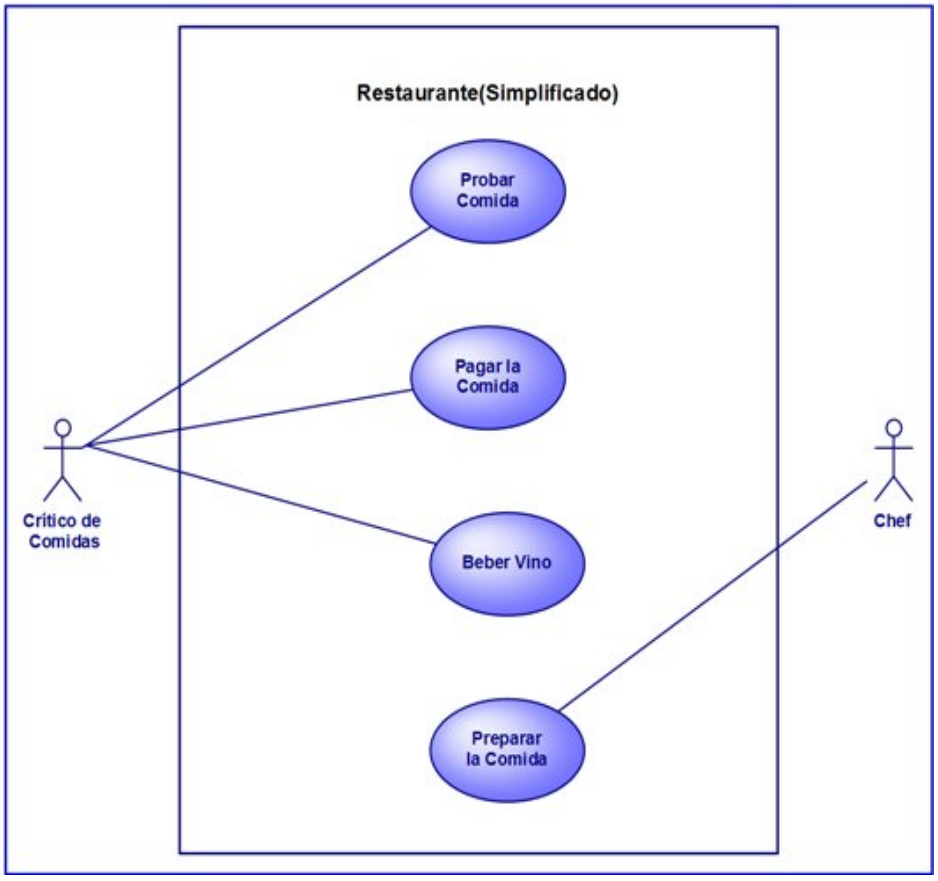


Figura 2.8 Representación de un Diagrama de Caso de Uso

Relación: conexión entre elementos del sistema.

Relación de inclusión: Una relación de inclusión de un caso de uso A, a un caso de uso B, indica que una instancia del caso de uso A contiene el comportamiento especificado por una instancia del caso de uso B. Se representa con una línea punteada entre los elementos y una flecha abierta puede indicar quien inicia la relación, sobre la línea debe ir la etiqueta <<include>> (Figura 2.9).

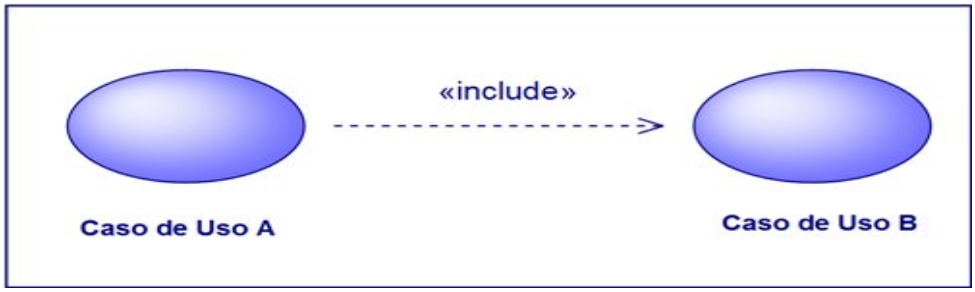


Figura 2.9 Representación de una Relación de Inclusión

Relación de extensión: Indica que un caso de uso base puede ser aumentado por un caso de uso de extensión, en caso de satisfacer una condición de extensión. Un caso de uso base define un punto de extensión, mientras que un caso de uso de extensión define la condición de extensión que debe ser satisfecha de manera que se inserte la extensión del caso de uso, en el caso de uso base. (Figura 2.10)

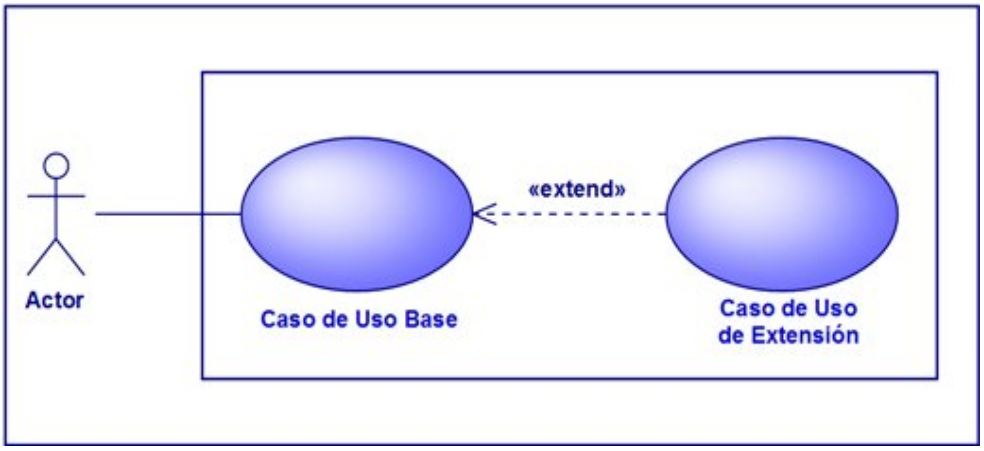


Figura 2.10 Representación de una Relación de Extensión

- **Diagrama de clases:** Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información manejada en el sistema, los componentes a encargarse del funcionamiento y la relación entre ellos. Un diagrama de clases, es un grafo de elementos clasificadores conectados por sus relaciones estáticas. (Figura 2.11). Estos diagramas son importantes para coordinar todos los requisitos que diferentes realizaciones de casos de uso imponen a una clase, a sus objetos y a los subsistemas que contiene.

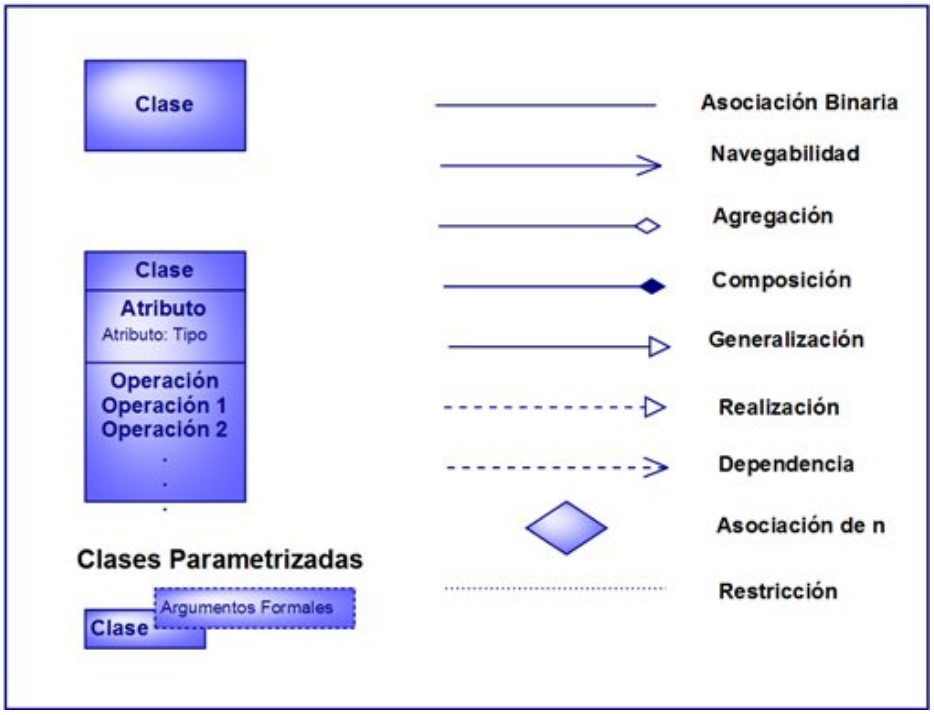


Figura 2.11 Elementos del Diagrama de Clases y Objetos.

- **Diagrama de secuencia:** Un diagrama de secuencia es un diagrama de interacción UML. Estos diagramas muestran la secuencia de mensajes que se van lanzando los objetos implicados en una determinada operación del programa. Dentro del diagrama los objetos se alinean en el eje X respetando su orden de aparición. En el eje Y se van mostrando los mensajes que se envían, también respetando su orden temporal (ver Figura 2.12).

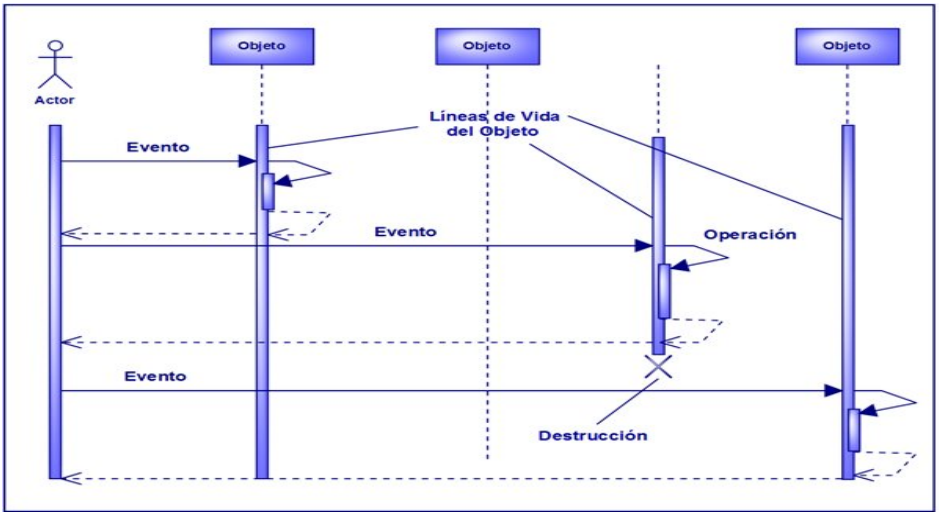


Figura 2.12 Componentes del Diagrama de Secuencia.

- **Diagrama de colaboración:** Un diagrama de colaboración es una forma alternativa al diagrama de secuencia de mostrar un escenario. Este tipo de diagrama muestra las interacciones entre objetos organizados en relación a los enlaces entre ellos. (Figura 2.13)

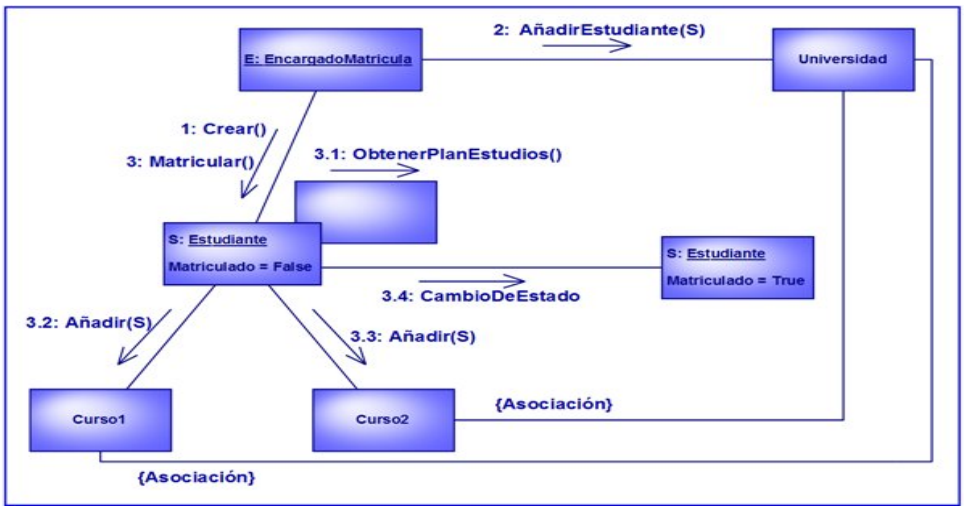


Figura 2.13 Diagrama de Colaboración

- **Diagrama de componentes**: Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes, etc.) y muestra las dependencias entre éstos. (Figura 2.14.)

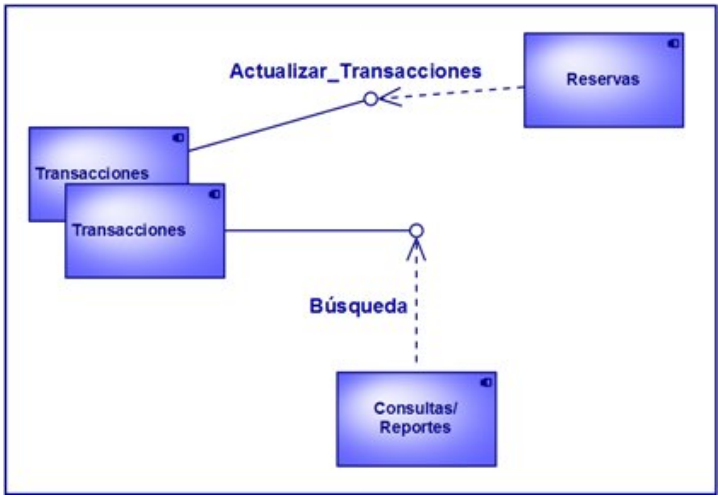


Figura 2.14 Diagrama de Componentes

- **Diagrama de despliegue:** El diagrama de despliegue es un tipo de diagrama del lenguaje unificado de modelado útil para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. (Figura 2.15) **[8]**

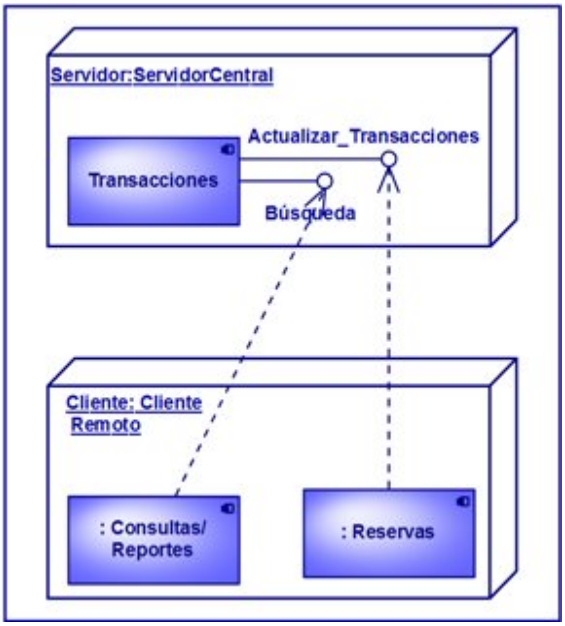


Figura 2.15 Diagrama de despliegue.

2.3 Programación Orientado a Objeto (POO)

La programación orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real en relación a otros tipos de programación.

La programación orientada a objetos como paradigma, "es una forma de pensar, una filosofía, de la cual surge una cultura nueva con técnicas y metodologías diferentes. La POO es una postura ontológica: el universo computacional está poblado por objetos, cada uno responsabilizándose por sí mismo, y comunicándose con los demás por medio de mensajes".

Se debe distinguir que la POO como paradigma (enfoque o manera de visualizar la realidad) y como metodología (colección de características para la ingeniería de software) no es lo mismo. Sin embargo, la publicidad asocia

la POO más a una metodología con respecto al paradigma. El interés en la POO radica más en los mecanismos aportados para la construcción de programas que en aprovechar un esquema alterno para el modelado de procesos computacionales.

Esta programación desde el punto de vista computacional, “es un método de implementación donde los programas son organizados como grupos cooperativos de objetos, representando una instancia de alguna clase, y éstas a su vez son miembros de una jerarquía de clases unidas por relaciones de herencia”. **[9]**

El paradigma orientado a objetos (OO) se basa en el concepto de objeto. Un objeto es aquello que tiene estado (propiedades más valores), comportamiento (acciones y reacciones a mensajes) e identidad (propiedad que lo distingue de los demás objetos). La estructura y comportamiento de objetos similares están definidos en su clase común; los términos instancia y objeto son intercambiables. Una clase es un conjunto de objetos compartiendo una estructura y un comportamiento común.

La diferencia entre un objeto y una clase es la siguiente: un objeto es una entidad concreta existente en tiempo y espacio; y una clase representa una abstracción, la "esencia" de un objeto. Un objeto no es una clase, pero una clase puede ser un objeto. Un modelo orientado a objetos contiene los siguientes elementos:

- **Abstracción:** Es una descripción simplificada o especificación de un sistema que enfatiza algunos de los detalles o propiedades del sistema, mientras suprime otros.

- **Encapsulación:** Es el proceso de ocultar todos los detalles de un objeto sin contribución a sus características esenciales.
- **Modularidad:** Es la propiedad de un sistema descompuesto en un conjunto de módulos coherentes e independientes.
- **Jerarquía o herencia:** Es el orden de las abstracciones organizadas por niveles.
- **Tipificación:** Es la definición precisa de un objeto de tal forma que objetos de diferentes tipos no puedan ser intercambiados o, cuando mucho, puedan intercambiarse de manera muy restringida.
- **Concurrencia:** Es la propiedad de distinguir un objeto activo de uno inactivo.
- **Persistencia:** Es la propiedad de un objeto a través de la cual su existencia trasciende en el tiempo (es decir, el objeto continúa existiendo después de que su creador ha dejado de existir) y/o el espacio (es decir, la localización del objeto se mueve del espacio de dirección en que fue creado).

Las relaciones entre objetos definen el comportamiento del sistema. Se dice que un objeto es un actor si su única función es operar sobre otros objetos. El objeto es un servidor si sólo es manejado por otros objetos y es un agente si tiene ambas propiedades. Los objetos actúan entre sí mediante mensajes, es decir, acciones pedidas por el objeto transmisor para ejecutar el objeto receptor. Dependiendo del comportamiento definido para un objeto, éste tomará las acciones para ejecutar o no el mensaje, de manera apropiada.

En cuanto a las metodologías OO, se dice que hay un gran número de métodos orientados a objetos actualmente. Muchos de los métodos pueden

ser clasificados como orientados a objetos porque soportan de manera central los conceptos de la orientación a objetos. **[9]**

2.4 Base de Datos

Es un conjunto exhaustivo de datos estructurados, fiables y homogéneos; organizados independientemente de su utilización y de su implementación en máquina, accesibles en tiempo real, compartibles por usuarios concurrentes con necesidades de información diferentes y no predecibles en el tiempo. Una base de datos es un conjunto de datos pertenecientes al mismo contexto almacenados sistemáticamente para su uso posterior.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. **[10]**

2.4.1 Sistema manejador de base de datos

Es una colección de numerosas rutinas de software relacionadas entre sí, cada una de las cuales es responsable de una tarea específica. El objetivo primordial de un sistema manejador base de datos es proporcionar un contorno conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la base de datos. Todas las peticiones de acceso a la base de datos, se manejan centralizadamente por medio del DBMS, por lo que este paquete funciona como interface entre los usuarios y la base de datos.

2.4.2 Esquema de base de datos

Es la estructura por la que está formada la base de datos, se especifica por medio de un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos. (DDL)

2.4.3 Administrador de base de datos (DBA)

Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos, generalmente tiene(n) experiencia en DBMS, diseño de bases de datos, sistemas operativos, comunicación de datos, hardware y programación.

2.5 Lenguaje de Programación Visual Basic

Es uno de los lenguajes de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Visual Basic es también un lenguaje basado en objetos, aunque no orientado a objetos como C++ o Java. La diferencia está en que Visual Basic utiliza objetos con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo propios de los verdaderos lenguajes orientados a objetos.

Existen distintos tipos de programas, entre ellos están: secuenciales, interactivos y orientados a eventos. En los primeros tiempos de los ordenadores, los programas eran de tipo secuencial (también llamados tipo

batch). Un programa secuencial se arranca, lee los datos que necesita, realiza los cálculos e imprime o guarda en el disco los resultados. Mientras este programa está ejecutándose, no se necesita ninguna intervención del usuario.

Los programas interactivos exigen la intervención del usuario en tiempo de ejecución para suministrar datos y/o indicar al programa lo que debe hacer por medio de menús. Estos programas limitan y orientan la acción del usuario.

Los programas orientados a eventos son los típicos de Windows, tales como Netscape, Word, Excel y Powerpoint. Cuando uno de ellos ha arrancado, lo único que hace es quedarse a la espera de las acciones del usuario (eventos). El usuario dice si quiere abrir y modificar un fichero existente, o bien comenzar a crear un fichero desde el principio. Éstos pasan la mayor parte de su tiempo esperando los eventos y respondiendo a ellos.

Visual Basic está orientado a la realización de programas para Windows, pudiendo incorporar todos los elementos de este entorno informático: ventanas, botones, caja de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, entre otros. **[11]**

Esta aplicación puede trabajar de 2 modos distintos: en modo de diseño y en modo de ejecución. En modo de diseño, el usuario construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades, y desarrollando funciones para gestionar los eventos. En modo de ejecución, el usuario actúa sobre el programa introduciendo eventos y prueba como responde el mismo. Hay algunas propiedades de los controles que deben establecerse en modo de diseño,

pero muchas otras pueden cambiarse en tiempo de ejecución desde el programa escrito en Visual Basic.

CAPÍTULO 3: FASE DE INICIO

3.1 Introducción

Para realizar este proyecto se escoge la metodología del Proceso Unificado, con el propósito de modelar, documentar y obtener un software de calidad. Esta metodología consta de cuatro fases: Inicio, elaboración, construcción y transición.

Durante este capítulo se abordará la primera fase del proceso unificado: La Fase de Inicio. Esta fase constituye el inicio del desarrollo del programa SAGEAV (Software que permita la Automatización para la Gestión de una Empresa Arrendadora de Vehículos), nombre con el cual será referenciado a lo largo de la realización de este trabajo.

3.2 Fase de Inicio

En la fase de inicio se da comienzo al Proceso Unificado de Desarrollo de Software; es donde se pone en marcha el proyecto a partir de una idea inicial, definiendo el ámbito del sistema y esbozando las arquitecturas candidatas; así como también la captura de requisitos, para la parte de análisis y diseño, indicando así que en la etapa se identifican y prevalecen los riesgos más importantes presentes en el proyecto.

3.3 Captura de Requisitos

Un requisito es una característica de diseño, una propiedad o un comportamiento de un sistema. Normalmente cualquier sistema interactúa con muchos usuarios, los cuales no tienen una visión global del mismo, no saben cómo puede hacerse más eficiente la operación en su conjunto y frecuentemente no saben cuáles son los requisitos, ni tampoco cómo especificarlos de una forma precisa. Por tal motivo la captura de requisitos por parte del desarrollador del software se hace complicada, los cuales tienen que realizar los fundamentos del flujo de trabajo relacionados con los requisitos, para guiar el desarrollo del proyecto hacia un sistema correcto. Este flujo de trabajo incluye los siguientes pasos:

3.3.1 Requisitos Candidatos

Los requisitos candidatos de un sistema son un conjunto de ideas que aparecen durante la vida de un sistema, por parte de los clientes, usuarios, Analistas y desarrolladores. Para la identificación de los mismos fue necesario realizar una serie de visitas, previamente planificadas a lo largo de la gerencia general, con el propósito de observar directamente las actividades que se realizan cotidianamente a la hora de alquilar un vehículo.
Recopilada toda esta información, se identificaron los siguientes requerimientos:

1. Elaboración del alquiler de vehículos de forma automatizada.
2. Generar un formato estándar para el arrendamiento de vehículo.
3. Almacenar los datos asociados al proceso de generar tanto reservas como alquiler de vehículos en una base de datos siguiendo los estándares de la empresa.

4. Permitir consultas a los datos almacenados.
5. El diseño de las interfaces y el código del programa deben estar dentro de los estándares de la empresa.

3.3.2 Requisitos Funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar o los aspectos que cada usuario quiere que el sistema haga, los cuales serán llevados por el Analista a casos de uso que representarán los diferentes modos en que el usuario puede utilizar el sistema. Dentro de los requerimientos funcionales exigidos para SAGEAV tenemos los siguientes:

- Proporcionar al personal encargado de alquilar vehículos, una interfaz que permita la visualización de las características de los vehículos que se reservarán o se alquilarán, haciéndolas lo más similar posible a los diseños de los otros software de control con los cuales cuenta la empresa **Oriente Rent-A-Car C.A.**

- Permitir al personal que genera el arrendamiento, tener notificación sobre el estatus en el cual se encuentra vehículo, es decir, registrando y dando un aviso en el sistema de la fecha y hora en la que fue alquilado el vehículo así como también a la hora que debe ser devuelto.

- Permitir al personal agregar las características que considere necesarias para identificar de forma correcta al vehículo que va a salir de la empresa.

- Permitir al usuario consultar datos tanto de los clientes como la de los vehículos registrados en el sistema, mostrando todas las características que se hayan guardado en la fecha seleccionada.

3.3.3 Requisitos No Funcionales

Define las características que de una u otra forma puedan limitar el sistema, encontrándose entre ellos:

- El requerimiento de los equipos instalados deben cubrir las necesidades que exige el software para su pleno funcionamiento (en tiempo y espacio).

- El software no puede ser manipulado por personal ajeno a la empresa **Oriente Rent-A-Car C.A**.

- El personal encargado del mantenimiento del programa debe conocer la aplicación y las necesidades del personal.

3.3.4 Contexto del Sistema

El contexto del sistema, se define con el firme conocimiento que debe tener el desarrollador del software acerca del entorno donde se desarrollará el sistema. Para comprender el contexto actual en el cual se desarrolla el proceso de arrendamiento de vehículos, se coordino conjuntamente con el personal encargado de alquilar vehículos, la revisión de toda la documentación disponible de los procedimientos implementados en **Oriente**

Rent-A-Car C.A., para poder controlar el proceso de gestión de alquilar vehículos, mediante formatos elaborados en hojas de texto Microsoft Word.

Al finalizar con todas las actividades mencionadas anteriormente se estableció el contexto en el cual se creará el software **"SAGEAV"** (Software que permita la Automatización para la Gestión de una Empresa Arrendadora de vehículos), logrando determinar la lista de requerimientos de información con los cuales debería cumplir.

3.3.4.1 Modelo de Dominio

El modelo de dominio, describe los conceptos importantes del contexto como objeto del dominio, el cual representa cosas que existen o eventos que suceden en el entorno del trabajo del sistema.

Muchos de los objetos del dominio o clases (para emplear una terminología más precisa) pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio.

En el Modelo de Dominio se debe obviar cualquier información acerca del comportamiento interno del sistema y de cómo soportará las funcionalidades esperadas, ya que tiene un fin de tratamiento y comprensión del sistema y no de la solución.

Este modelo se describe mediante diagramas de UML, especialmente diseñados para tal fin, como es el diagrama de clases de análisis.

3.3.4.2 Diagrama de Clases de Dominio

La interacción entre las clases del dominio se ha estructurado en un diagrama que permite establecer las relaciones entre éstas. La Figura 3.1, describe el diagrama de modelo de clase de dominio del proyecto en desarrollo, donde se permite establecer las relaciones y asociaciones que existen entre cada uno de las clases anteriormente descritas. La definición de los diversos elementos que conforman los modelos de dominio los podemos detallar en la Tabla 3.1.

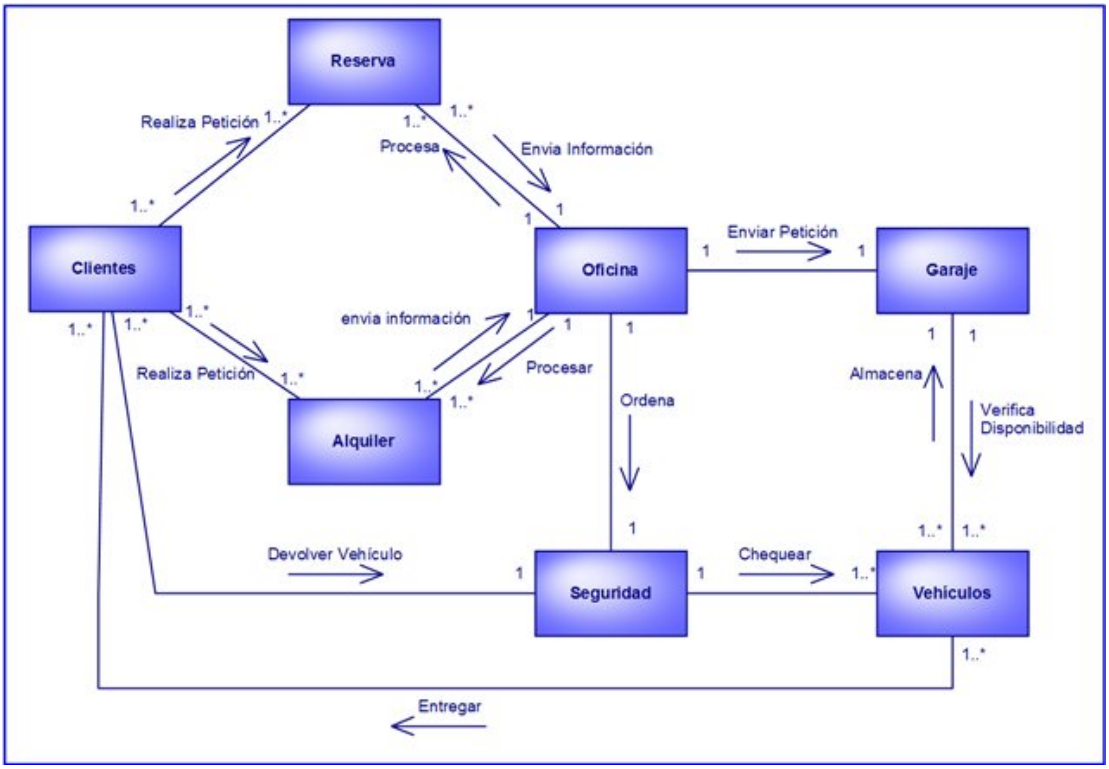


Figura 3.1 Diagrama de clases de Dominio.

Tabla 3.1 Descripción de las Clases de dominio

Clases de dominio	Descripción
Clientes	Representa a las personas interesadas en solicitar el arrendamiento de vehículos.
Seguridad	Representa el puesto de vigilancia, donde se verifican y revisan los vehículos que entran y salen de la empresa.
Oficina	Es la encargada de recibir peticiones tanto de alquiler como de reserva de vehículos, además dichas peticiones son enviadas al garaje en donde a su vez, se hace la verificación de los vehículos que están disponibles, para así continuar con el proceso de alquilar o reservar un vehículo.
Garaje	Representa el Departamento de la empresa en donde se almacenan los vehículos de la empresa **Oriente Rent-A-Car C.A.**
Vehículos	Representa la identificación de todos los vehículos que se encuentran disponibles en el Garaje.
Alquiler	Representa la parte en cual la oficina departamental gestiona cualquier tipo de solicitud de alquiler de vehículo que realicen los clientes.
Reserva	Representa la parte en cual la oficina departamental gestiona cualquier solicitud de reservación de vehículos que realicen los clientes.

3.3.5 Identificación y Descripción de Riesgos

Un riesgo es una probabilidad de que el proyecto se vea afectado en su desarrollo o posteriormente en su comportamiento y deben ser tratados de acuerdo a su importancia o prioridad.

Durante esta fase (**inicio**) se identifican los riesgos críticos, donde el desarrollador intentará mitigarlos, explorando su naturaleza hasta el punto de preparar un plan de iteraciones, para disminuir el impacto en la planificación, el costo o la calidad del proyecto.

Los riesgos se pueden clasificar en:

- **Riesgos específicos de un producto particular:** Estos riesgos están referidos a problemas que podrían surgir de las técnicas y herramientas que se piensan utilizar para abordar los requerimientos durante la implementación, o del comportamiento del sistema ante ciertas entradas o exigencias.

- **Riesgo de no conseguir la arquitectura correcta:** Uno de los riesgos más serios es el de no construir un sistema que pueda evolucionar suavemente por las fases siguientes o durante su tiempo de vida, es decir, el no establecer una arquitectura flexible.

- **Riesgo de no conseguir los requisitos correctos:** Constituye el riesgo de no construir un sistema que haga lo que los usuarios quieren realmente que haga. Este tipo de riesgo depende en gran medida del trabajo que se haga durante la captura de requisitos.

Una vez que se han identificado los riesgos, se procede a manipularlos de varias formas. Se cuenta fundamentalmente con cuatro elecciones: evitarlo, eliminarlo, atenuarlo o controlarlo. Algunos riesgos pueden o deberían evitarse, quizás mediante una planificación del proyecto o un cambio en los requisitos, otros deberían limitarse, es decir, restringirse de modo que solo afecten a una parte del proyecto, otros riesgos pueden atenuarse ejercitándolos y observando si aparecen o no. Lo único que se puede hacer es controlarlo y observar si aparecen. Si esto ocurre, se debe tener un plan de contingencia.

Para la identificación de riesgos se realizó la observación directa, revisión bibliográfica, consulta de documentos disponibles dentro de la empresa y entrevistas informales. A través de estas técnicas se logró obtener la información necesaria para realizar un análisis que permitiera identificar los principales riesgos que podrían limitar el desarrollo del software.

Para precisar si el software era factible, se identificaron los riesgos críticos y a partir de ellos se diseñaron los casos de uso principales del software, para de esta forma poder tratar y eliminar dichos riesgos.

Entre los riesgos críticos que se pueden presentar en el desarrollo de **SAGEAV**, se pueden mencionar los siguientes:

- **Falta de dominio del contexto**

  - **Descripción:** Este riesgo reside en la dificultad de que no se pueda tener dominio completo sobre el manejo del sistema actual.

- **Prioridad:** Crítico.
- **Responsable:** Desarrollador.
- **Impacto:** Funcionalidad del sistema.
- **Contingencia:** Analizar las herramientas disponibles en la empresa para verificar la vialidad de ellas en el desarrollo del software.

- **Empleo de herramientas no apropiadas**

  - **Descripción:** Este riesgo reside en la dificultad que pueda involucrar el uso de un lenguaje de programación o herramienta que no brinden el soporte adecuado al desarrollo del sistema.
  - **Prioridad:** Crítico.
  - **Responsable:** Desarrollador.
  - **Impacto:** Funcionalidad del sistema.
  - **Contingencia:** Analizar las herramientas disponibles en la empresa para verificar la vialidad de ellas en el desarrollo del software.

- **Fallas en la conexión a la base de datos**

  - **Descripción:** Este riesgo reside en la posibilidad de que se produzca fallas al momento de codificar el acceso a la base de datos y no utilizar los componentes adecuados para ello.
  - **Prioridad:** Crítico.
  - **Responsable:** Desarrollador.
  - **Impacto:** Funcionalidad del sistema.

- **Contingencia:** Revisar la documentación relacionada con el acceso a base de datos a través del código y componentes.

- **No conseguir requisitos correctos**

  - **Descripción:** Se desea construir un sistema que cubra las necesidades que exige el usuario y de la forma que éste espera. No tiene sentido desarrollar un sistema que no satisfaga los requerimientos del cliente o lo haga de forma incorrecta.
  - **Prioridad:** Crítico.
  - **Responsable:** Desarrollador.
  - **Impacto:** Funcionalidad del sistema.
  - **Contingencia:** Analizar en profundidad el sistema.

- **Configuración del sistema**

  - **Descripción:** Este riesgo consiste en la administración general de todos los términos asociados al sistema, necesarios para el funcionamiento del mismo.
  - **Prioridad:** Crítico.
  - **Responsable:** Desarrollador.
  - **Impacto:** Todo el sistema.
  - **Contingencia:** Se deben crear módulos que permitan la gestión de la información contenida en la base de datos: Inserción, eliminación, consulta y modificación.

- **No conseguir la arquitectura correcta**

  - **Descripción:** Es necesario conseguir una arquitectura software que permita la evolución del sistema por sus diferentes fases y durante su tiempo de vida.
  - **Prioridad:** Crítico.
  - **Responsable:** Desarrollador.
  - **Impacto:** Arquitectura del sistema.
  - **Contingencia:** Analizar en profundidad el sistema y estudiar en detalle las fases del Proceso Unificado de Desarrollo de Software.

- **Emisión de reportes**

  - **Descripción:** Se desea construir un sistema que sea capaz de emitir reportes impresos.
  - **Prioridad:** Secundario.
  - **Responsable:** Desarrollador.
  - **Impacto:** Reportes del sistema.
  - **Contingencia:** Estudiar la posibilidad de conectar el sistema con aplicaciones que brinden soporte de impresión.

Como se puede observar, cada riesgo, está intrínsecamente asociado con una fracción de la funcionalidad del sistema.

3.4 Diagrama de Casos de Uso

Los casos de uso han sido adoptados casi universalmente para la captura de requisitos de software en general, durante el desarrollo de este trabajo de grado tomaran una gran relevancia ya que esta herramienta se utilizará para dirigir el proceso de desarrollo en su totalidad.

Normalmente, un sistema posee muchos tipos de usuario. Cada usuario es representado por un actor. Los actores utilizan el sistema interactuando con los casos de usos.

Un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer un resultado observable para un actor. De los requerimientos que maneja la aplicación SAGEAV se identificaron como casos de uso, los siguientes (Ver Tabla 3.2):

Tabla 3.2 Casos de Uso identificados

Caso de uso	Descripción	Actor(es)
Acceso al Sistema	Permite al administrador la posibilidad de aceptar o denegar el acceso de otro usuario. El sistema presenta una ventana donde requiere el nombre de usuario y clave para poder acceder a él. Esta información se almacena en la base de datos del sistema mediante el Manejador de Bases de Datos.	Usuario, Manejador de Base de Datos
Gestionar Clientes	Corresponde al proceso de identificar y modificar el estatus de los clientes que estén en la lista de reservas, así como también de aquellos que se encuentren de alta o baja de los vehículos que hayan sido alquilados.	Usuario, Manejador de Base de Datos

Gestionar Vehículos	Este proceso se encarga de identificar y modificar el estatus de los vehículos que estén en lista de reservas, así como también el conjunto de características que posee cada vehículo.	Usuario, Manejador de Base de Datos
Gestionar Arrendamiento	Este proceso se encarga de verificar y actualizar el estatus de los vehículos que estén alquilados, se encuentren en reserva, o estén en mantenimiento.	Usuario, Manejador de Base de Datos
Realizar Consultas	Este proceso comprende las diversas combinaciones de consultas útiles para el control de vehículos y/o clientes.	Usuario, Manejador de Base de Datos
Solicitar Reporte	Este proceso se encarga de generar los reportes impresos realizados por el sistema para el analista, los cuales permiten tener disponible la información necesaria para la toma de decisiones.	Usuario, Manejador de Base de Datos
Gestionar Usuarios	Este proceso controla la completa gestión de los usuarios SAGEAV.	Administrador, Manejador de Base de Datos
Configuración	Este proceso maneja las diferentes variables de configuración existentes en el sistema.	Administrador, Archivos de Configuración
Mostrar Ayuda	Se encarga del manejo de la ayuda necesaria para la asistencia de los usuarios.	Usuario, Archivos de Ayuda

Con la información recopilada en la tabla 3.2 podemos establecer nuestro diagrama de casos de uso, el mismo se visualizaría de la siguiente manera (ver Figura 3.2):

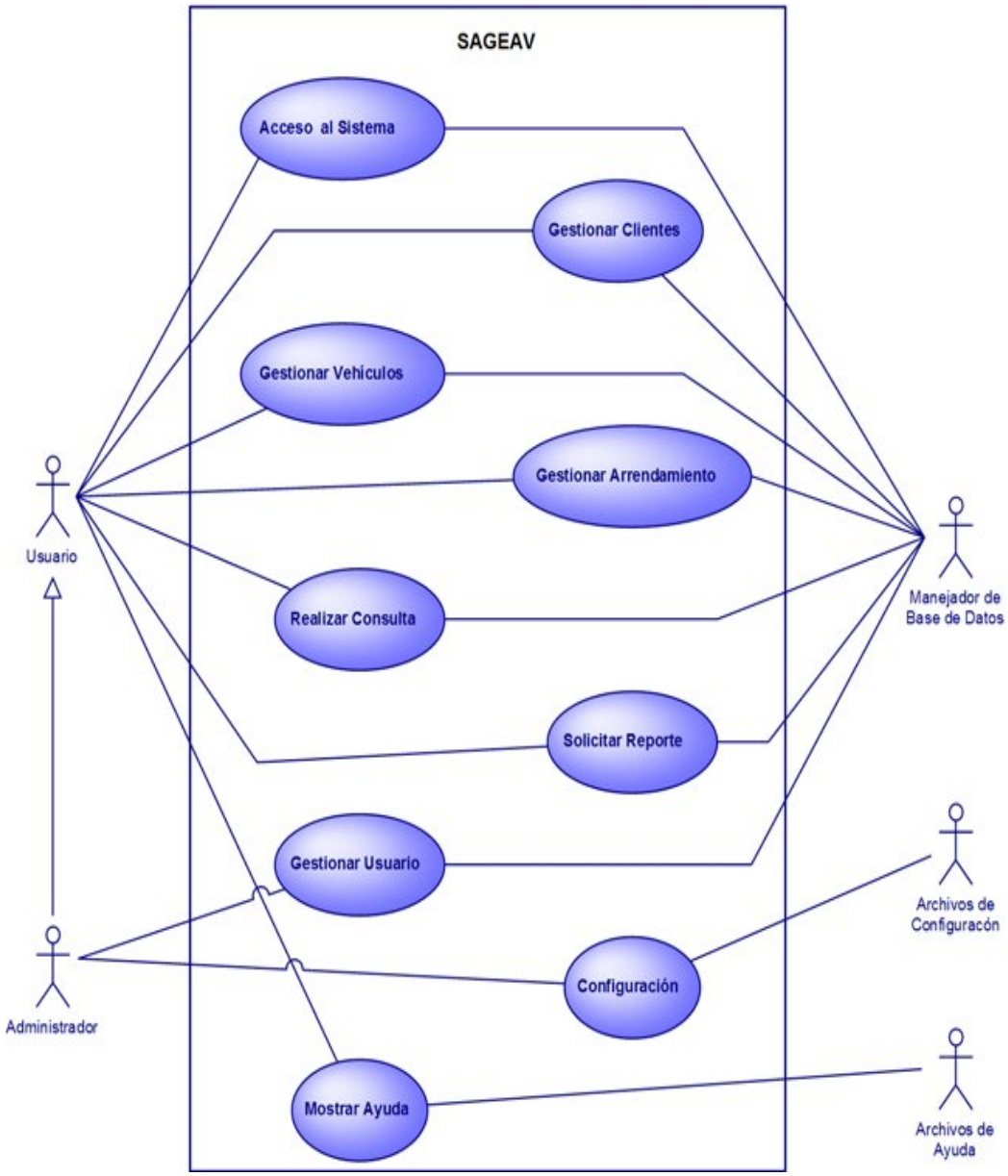


Figura 3.2 Diagrama de caso de uso general del sistema SAGEAV.

3.4.1 Descripción del caso de uso general del sistema SAGEAV

Mediante este diagrama de Caso de uso se muestran el conjunto de acciones que el software llevará a cabo para satisfacer las peticiones de los actores del sistema. El Administrador es un usuario con un mayor nivel de prioridad, el mismo podrá ejecutar todos los casos de usos y como labor exclusiva se encargará de configurar el sistema cuando sea necesario además del manejo de los usuarios que hacen uso de la aplicación.

El caso de uso “Configuración” se encargará de gestionar los cambios o actualizaciones que el sistema requiera para funcionar eficientemente.

Entre las acciones necesarias para el control de los vehículos se requiere el diseño de módulos como lo son: “Gestionar Vehículos” encargado de manipular la estructura detallada de la información de todos los vehículos que han sido ingresado en el sistema, “Gestionar Arrendamiento” que facilita el manejo de los datos asociados a los estatus de los vehículos, “Gestionar Clientes” que permite la manipulación de los datos asociados a todos los clientes que se encuentran almacenado en sistema, “Realizar Consulta” que facilitara al usuario la visualización del estatus tanto de los clientes como de los vehículos así como cualquier arrendamiento asociado, “Solicitar Reporte” encargado de la manipulación de los diversos reportes de los estatus de los vehículos registrados por los usuarios en el sistema.

El caso de uso “Mostrar Ayuda” representa un mecanismo muy importante que permitirá mostrar información al operador, para que éste pueda utilizar fácilmente el sistema.

El actor Manejador de Base de Datos se vincula con la mayoría de todos los escenarios del diagrama; esto es debido a que el mismo registra toda la información manejada por cada una de las interacciones de los procesos.

3.4.2 Descripción del caso de uso “Gestionar Clientes”

El presente caso de uso (ver Figura 3.3) se encarga de ingresar, modificar o eliminar datos de los clientes en el sistema. Por medio del proceso “Registrar Cliente” se permite al usuario añadir un nuevo cliente al sistema, “Modificar Cliente” el usuario tendrá la posibilidad de actualizar los datos de cada uno de los clientes existentes en el sistema, “Eliminar Cliente” el operador eliminará del sistema todo tipo de información relacionada a cualquier cliente.

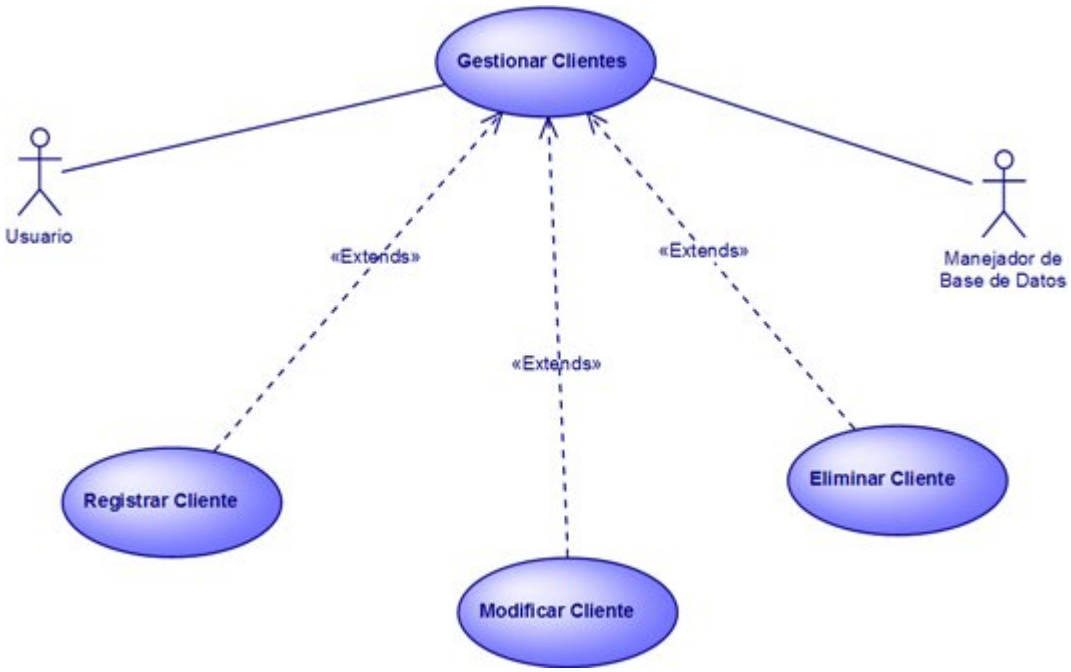


Figura 3.3 caso de uso “Gestionar Clientes”

3.4.3 Descripción del caso de uso “Gestionar Vehículos”

El presente caso de uso (ver Figura 3.4) se encarga de ingresar, modificar o eliminar los datos de todos aquellos vehículos que se encuentren en el sistema. Por medio del proceso “Agregar Vehículo” se permite al usuario introducir los datos de un nuevo vehículo al sistema, “Modificar Vehículo” el usuario tendrá la posibilidad de modificar cualquier tipo de datos de cada uno de los vehículos existentes en el sistema, “Eliminar Vehículo” el operador eliminará del sistema todo tipo de información relacionada a los vehículos de la empresa, “Asignar Mantenimiento” el usuario lo utiliza para registrar el control de los mantenimientos asignados a un vehículo.

.

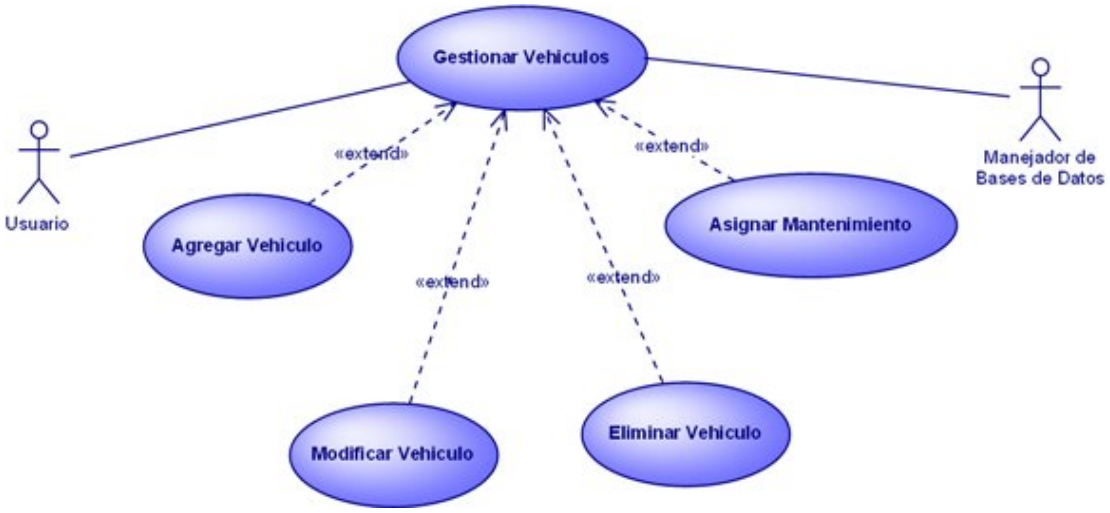


Figura 3.4 caso de uso “Gestionar Vehículos”

3.4.4 Descripción del caso de uso “Gestionar Arrendamiento”

En el presente caso de uso (ver Figura 3.5) el usuario se encarga del procesar los datos de los clientes que desean reservar o alquilar vehículos con su respectivo estatus. Por medio del proceso “Reservar Vehículo” se permite al usuario introducir los datos del cliente que solicita la reservación de un vehículo en el sistema, “Alquiler Vehículo” el usuario se encargará de gestionar todo tramite del cliente, respecto al proceso de alquiler de vehículo en el sistema. La posibilidad de modificar cualquier tipo de datos de cada uno de los vehículos existentes en el sistema, “Recibir Vehículo” el operador introducirá en el sistema todos los detalles de los vehículos que son recibidos por la empresa una vez de culminado el alquiler.

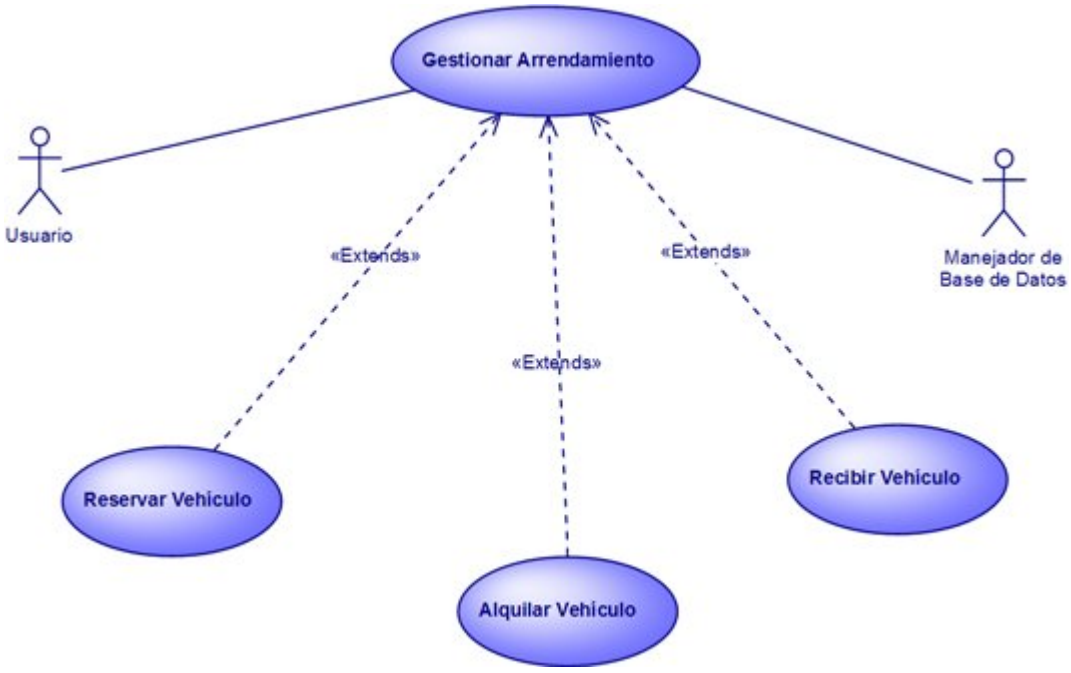


Figura 3.5 caso de uso “Gestionar Arrendamiento”

3.4.5 Descripción del caso de uso “Realizar Consulta”

El presente caso de uso (ver Figura 3.6) tiene como objetivo facilitarle al usuario de forma clara y concreta la información de mayor relevancia que permita hacer un seguimiento a los clientes con relación a los vehículos. Por medio del proceso “Consultar Datos Clientes ” el usuario logra visualizar todas las características propias de cada cliente, esto comprende la información relacionada con los vehículos, bien pudiera ser por consulta de clientes por reserva, por fecha de reserva, por fecha de alquiler, “Consultar Datos Vehículos” el usuario podrá observar todas las características de cada vehículo con relación a los clientes, el estatus de cada vehículo, así como los vehículos que estén en reserva o en mantenimiento, “Consultar Datos Arrendamiento” el usuario podrá observar el estatus de todos aquellos vehículos que se encuentren alquilados.

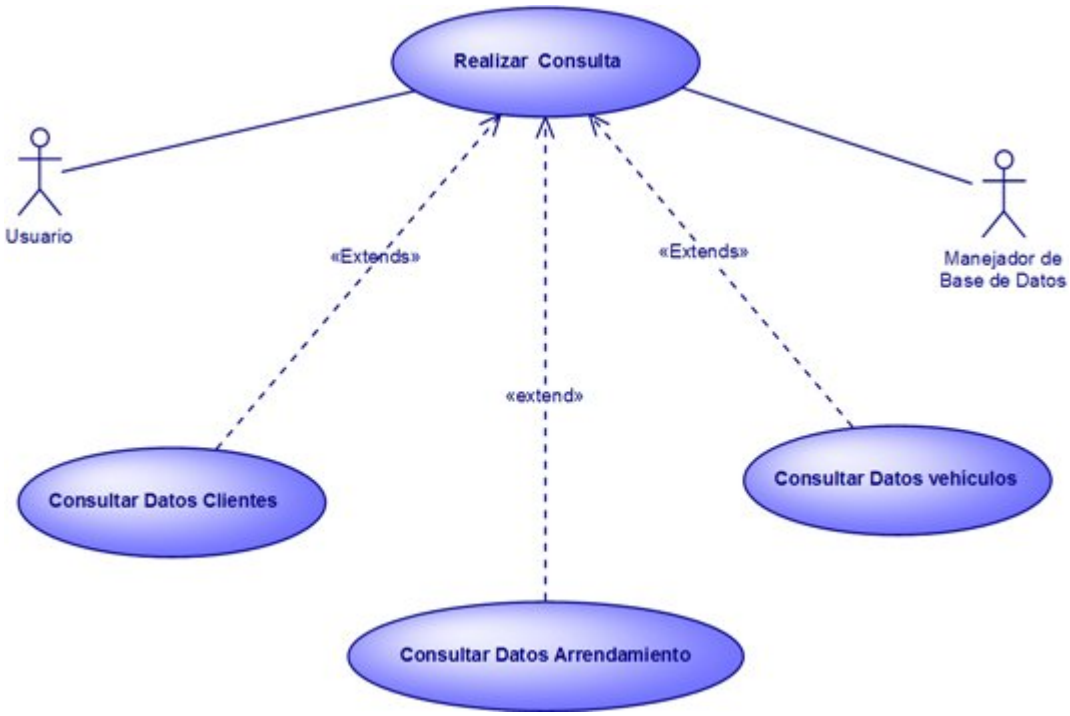


Figura 3.6 caso de uso “Realizar Consulta”

3.4.6 Descripción del caso de uso “Solicitar Reporte”

El caso de uso “Solicitar Reporte” (ver Figura 3.7) permite a los usuarios del sistema el manejo de los reportes tanto de los clientes como de los vehículos que deben ser anexados a los equipos cuando ocurriese algún desperfecto. El módulo “Generar Reporte de Mantenimiento” permite ingresar al sistema un nuevo reporte de mantenimiento ocurrido a cualquier vehículo de la empresa, además de la descripción del mismo, a través del proceso “Generar Reporte de Facturación” se logra la posibilidad de que el usuario pueda Factura de todos los vehículos que fueron arrendados en un cierto período de tiempo, bien sea diarios, semanales, mensuales o un rango definido por el mismo.

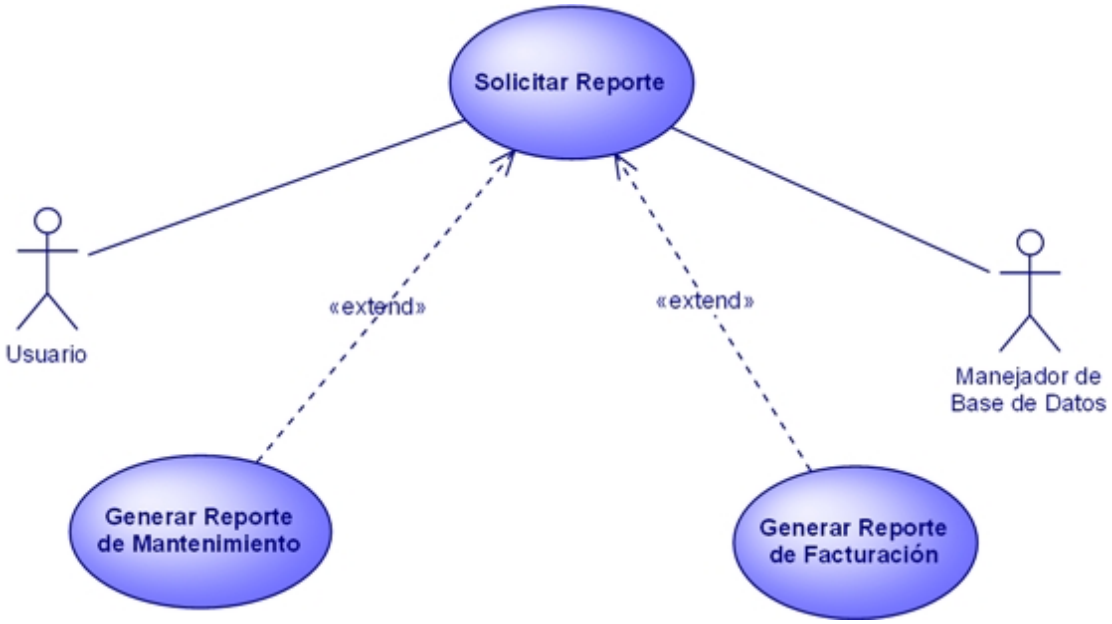


Figura 3.7 caso de uso “Solicitar Reporte”

3.5 Diagrama de Clase de Análisis

Los diagramas de clases de análisis se utilizan para representar y modelar los atributos y asociaciones que poseen las clases, bien sea del sistema en general o de un caso de uso en concreto. Para desarrollar estos diagramas se utilizan los estereotipos de interfaz, de control y de entidad, dichos elementos permiten representar de una manera sencilla los procesos.

El diagrama de clases de análisis para el caso de uso “Gestionar Arrendamiento” se puede observar en la Figura 3.8, en este se describe la forma de tramitar el proceso de alquiler de vehículos a todos los clientes que hacen vida en el sistema, de igual manera se muestran las clases que posteriormente en la etapa de diseño se conceptualizaran para su implementación en el sistema; para modelar los distintos casos de usos se utilizan tres estereotipos diferentes que representan las clases: clase de interfaz, clase de control y clase de entidad.

Las clases Interfaz permiten la interacción entre el sistema y los usuarios, éstas se representan en la Figura 3.8 mediante la IU: Reservar Vehículo, IU: Alquilar Vehículo e IU: Recibir Vehículo.

Las clases de Control representan la ejecución de acciones, coordinación, secuencia, procesos y control de otros objetos y se representan en el diagrama mediante: Capturar Data de Interfaz, Ingresar Reserva, Iniciar Alquiler, Procesar Devolución, Guardar Datos.

Y finalmente las clases Entidad modelan información, las cuales suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué depende el sistema, como se puede observar en la Figura 3.8. La clase

“Entidad” está representada por la clase “Arrendamiento” la cual representa toda la información almacenada necesaria para la completa gestión de los alquileres de autos.

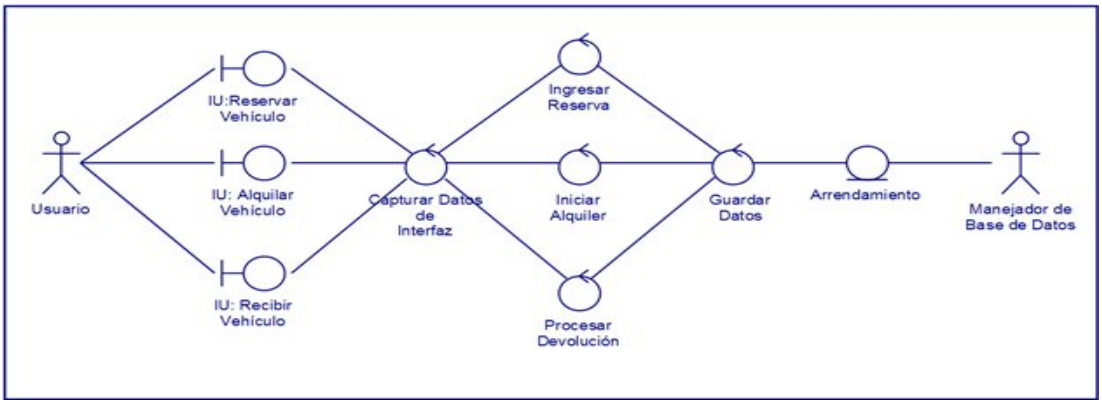


Figura 3.8 Diagrama de clase de análisis “Gestionar Arrendamiento”

3.6 DIAGRAMA DE COLABORACIÓN

Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Como se podrán observar los diagramas de colaboración poseen la misma estructura de los diagramas de clase de análisis, solo que ahora se representa la interacción entre estos componentes, y para ello se utilizan los mensajes, mediante un número se pretende organizar dicho mensajes y las flechas indican la dirección del flujo.

A continuación se muestran los diagramas de Colaboración de análisis desarrollados para el caso de uso “Gestionar Arrendamiento” (ver Figura 3.9)

con la correspondiente leyenda de los mensajes enviados entre los distintos elementos (ver Tabla 3.3):

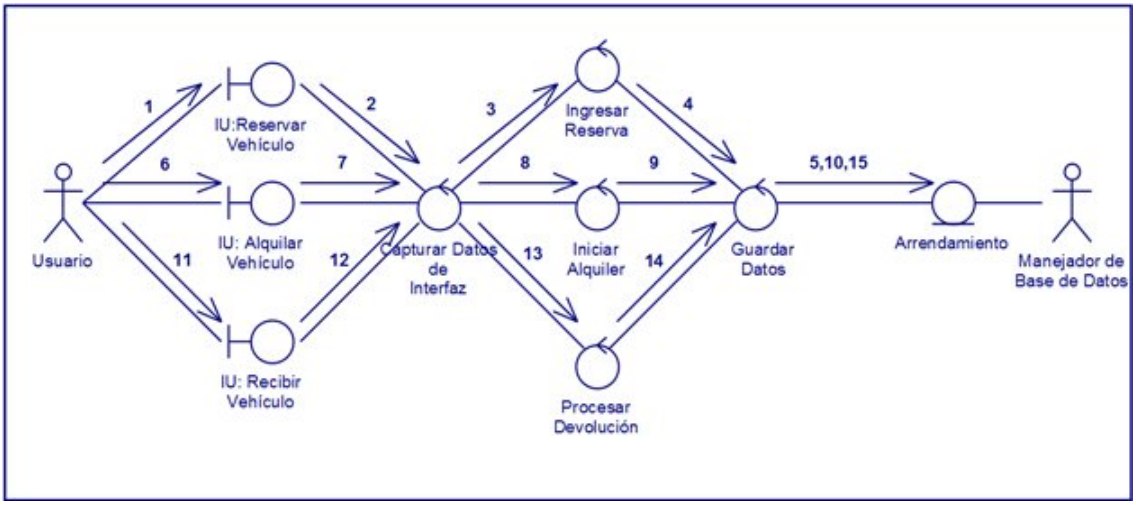


Figura 3.9 Diagrama de Colaboración "Gestionar Arrendamiento"

Tabla 3.3 Leyenda del Diagrama de colaboración "Gestionar Arrendamiento"

LEYENDA
1: Solicitar reservar vehículo.
2: Capturar datos ingresados por el usuario.
3: Asignar parámetros para introducir reservaciones.
4: Crear instancia de la reservación a ingresar.
5: Almacenar en la base de datos la reservación realizada.
6: Solicitar Alquilar vehículo.
7: Capturar datos introducido por el usuario.
8: Iniciar el proceso del alquiler del vehículo.
9: Crear instancia del alquiler a iniciar.
10: Almacenar en la base de datos el alquiler realizado.
11: Solicitar recibir vehículo
12: Capturar datos ingresados por el usuario.

13: Iniciar el proceso de devolución.
14: Crear instancia de la devolución de vehículo.
15: Almacenar en la base de datos los vehículos que fueron devueltos

3.7 Diagrama de Paquetes

Los paquetes se representan mediante rectángulos con pestañas y las relaciones de dependencias se muestran como flechas con líneas discontinuas.

Para el sistema desarrollado se identificaron una serie de paquetes, los cuales fueron agrupados y relacionados, obteniéndose de esta manera el artefacto llamado Diagrama de Paquetes de análisis del sistema.

El diagrama de la Figura 3.10 muestra los paquetes que encapsulan los diferentes casos de usos que fueron definidos al realizar el análisis del sistema, como se puede notar existen los paquetes “Realizar Consultas”, “Acceso al Sistema”, “Gestionar Clientes”, “Gestionar Vehículos”, “Gestionar Arrendamiento”, “Solicitar Reporte”, “Gestionar Usuarios” y “Configuración”.

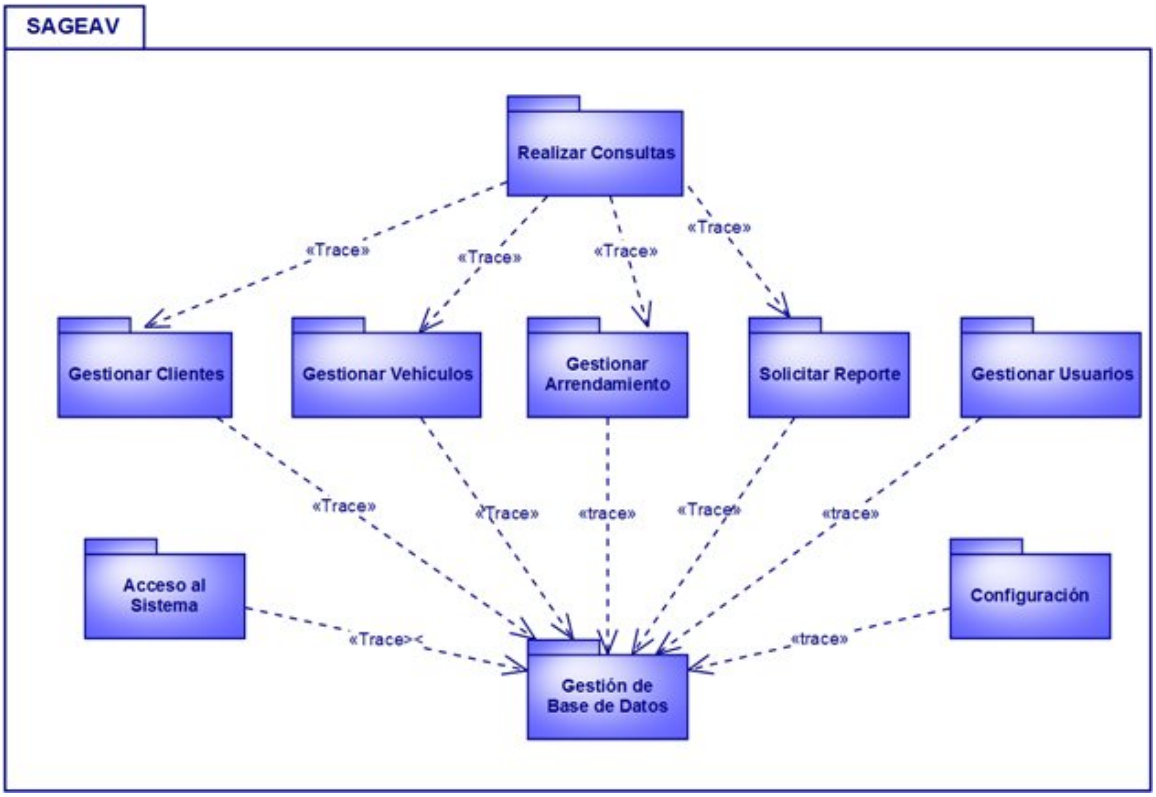


Figura 3.10 Diagrama de Paquetes de SAGEAV

3.8 Resumen de Fase de Inicio

En el sistema desarrollado se crearon artefactos que definieron de una manera clara la estructura del sistema, este trabajo facilito su posterior desarrollo. Con la realización del modelo de dominio del sistema presentado, se alcanzo de forma notable la comprensión del contexto del sistema. De esta misma manera se lograron identificar los actores principales, tanto humanos como sistemas externos y la relación de éstos en el sistema "SAGEAV", además se lograron establecer las tareas que cumplirían cada uno de estos elementos.

Mediante la representación del sistema por medio del diagrama de casos de uso se logro estructurar la aplicación en módulos que se encargan de cumplir con cada uno de los requerimientos del sistema detectados y por medio de los diagramas de clase de análisis y colaboración se alcanzo un estudio más detallado de un caso de uso en particular como lo es “Gestionar Arrendamiento”.

Para finalizar a través del encapsulamiento de los diversos casos de uso detectados utilizando para su representación el diagrama de paquetes se obtuvo una abstracción más clara de los diversos componentes de software que harán vida en el sistema y serán de utilidad para las posteriores etapas de desarrollo en este trabajo de grado.

CAPÍTULO 4: FASE DE ELABORACIÓN

4.1 Fase de Elaboración

La etapa de elaboración es muy importante en el desarrollo de este proyecto, ya que nos permitirá establecer la arquitectura sólida del software, la cual guiará el trabajo durante la fase de construcción y transición, así como también en las posteriores generaciones del proyecto.

El objetivo principal en esta fase, es formular la línea base de la arquitectura, cumpliendo con los flujos de trabajo de requisito, análisis, diseño e implementación.

En la fase de inicio se logró conceptuar el sistema, se identificó su contexto, sus riesgos críticos así como también la mayoría de los requisitos funcionales y no funcionales, los cuales suministran los insumos de la fase siguiente.

El fundamento de la fase de elaboración es desarrollar el flujo de trabajo del diseño, sustentado en la captura de requisitos, en el análisis y diseño preliminar; se evalúan y depuran los requisitos de la fase anterior considerando la creación de nuevos casos de uso, los cuales serán sujetos al análisis y al diseño, empezando por el esbozo de una arquitectura inicial para finalmente definir la arquitectura final del software.

En esta etapa se estima la creación de la interfaz de usuario y la elaboración de un buen diseño de las bases de datos, ya que el proyecto manipula bases de datos robustas que manejan gran cantidad de información, por tal motivo consideramos la evaluación de estructuras de

datos con la finalidad de procesar la información eficientemente. De no emplear adecuadamente una estructura de datos las consecuencias serian un retraso en el procesamiento de información y el objetivo principal de esta aplicación es agilizar el procesamiento de información en un tiempo considerablemente corto.

Consideramos desarrollar el flujo de datos de implementación y prueba en la fase de Construcción, ya que los mismos no tienen relevancia con el objetivo principal de la etapa de elaboración.

4.2 Diagrama de Clase de Diseño

Uno de los fundamentos del proceso unificado es que esta basado en la arquitectura del software, es muy importante entonces, estructurar correctamente dicha arquitectura, en la fase de inicio, mediante los diagramas de clase de análisis, se definen las posibles clases necesarias para llevar a cabo los casos de usos; ahora en la fase de elaboración partiendo de análisis previos definiremos mediante diagramas la arquitectura del software, el diseño del software es equivalente al bosquejo de la estructura de un edificio para un ingeniero civil, un ingeniero de software debe contar con un mecanismo que le permita representar los componentes fundamentales del sistema que está desarrollando, así como las relaciones que existen entre estos componentes, existen varios diagramas que se utilizan para representar dicha estructura, y uno de los más importantes es el diagrama de clase de diseño, en este se representaran las clases, que en adelante instanciarán los objetos. En la Figura 4.1 correspondiente al diagrama de clase de diseño del sistema SAGEAV se observa que las clases se representan mediante rectángulos y las relaciones se representan a través de las líneas que las conectan a dichas clases, también se nota el uso

de ciertos conectores que sirven para identificar la relación entre las mismas. Las clases fundamentales para este sistema son: Clientes, Vehículos, Arrendamiento, Consultas, Reportes, Usuarios, estas clases encapsulan, no solo las variables asociadas sino también los procedimientos que se llevaran a cabo dichas variables. Así mismo se representan las clases de interfaz que permiten comunicar al sistema con los entes externos, y los gestores de bases de datos que sirven para coleccionar la data requerida además de contener mecanismos internos para su gestión.

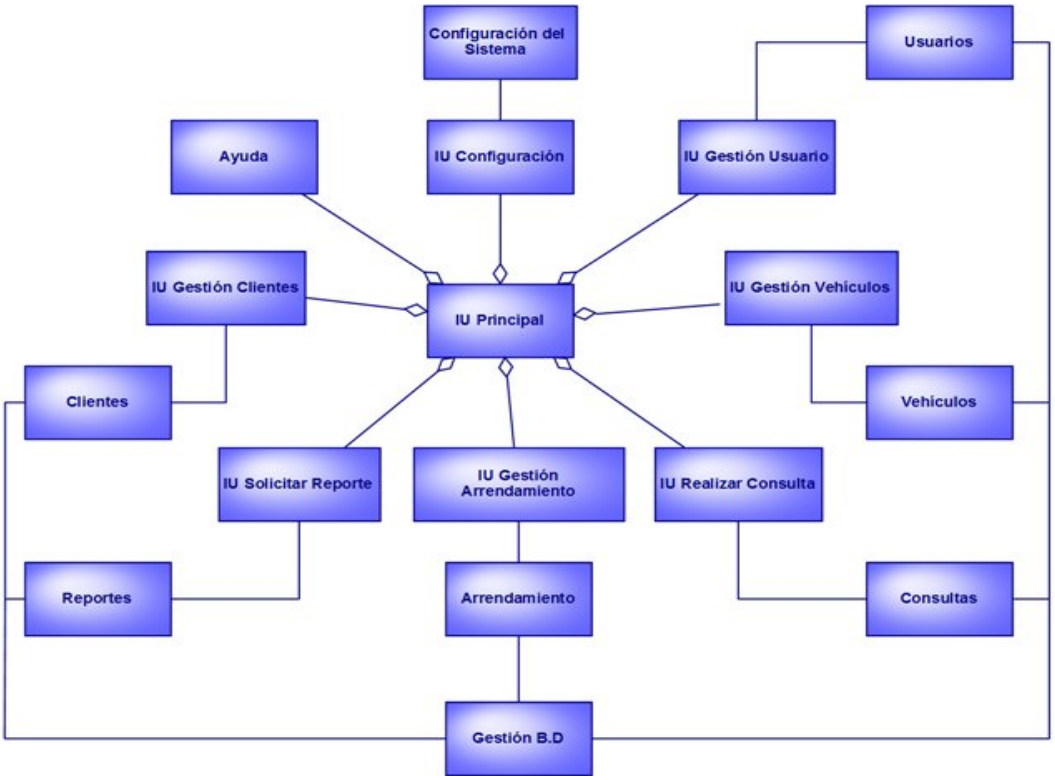


Figura 4.1 Diagrama de Clase de Diseño de SAGEAV.

4.3 Diagrama de Capas

El diagrama de capas es un diagrama que permite representar las diferentes capas de software que contendrá el sistema SAGEAV.

El diseño de la arquitectura del sistema involucra cuatro capas: la capa específica de la aplicación, la capa general de la aplicación, la capa intermedia y finalmente la capa de software del sistema.

El diagrama de capas mostrado en la siguiente figura (Figura 4.2) representa todas las capas del software en las que se distribuye la funcionalidad del sistema y que se requieren para que se lleve a cabo el sistema y en un futuro sea implementado.

En cada capa del diagrama se muestran las relaciones existentes entre los paquetes mediante las trazas. Se puede observar el paquete “Microsoft Windows”, que representa el sistema operativo usado durante su desarrollo, las herramientas de software: “Visual Basic 2005”, el sistema manejador de base de datos “SQL Server 2005”. En la capa general de la aplicación podemos apreciar los paquetes “Gestión Clientes”, “Gestión Vehículos”, “Gestión Arrendamiento”, “Realizar Consulta” y “Solicitar Reporte”, “Gestión de Base de Datos”. En la capa correspondiente la capa específica de la aplicación podemos apreciar los paquetes encargados de la “Gestión de usuarios” y “Configuración del sistema”.

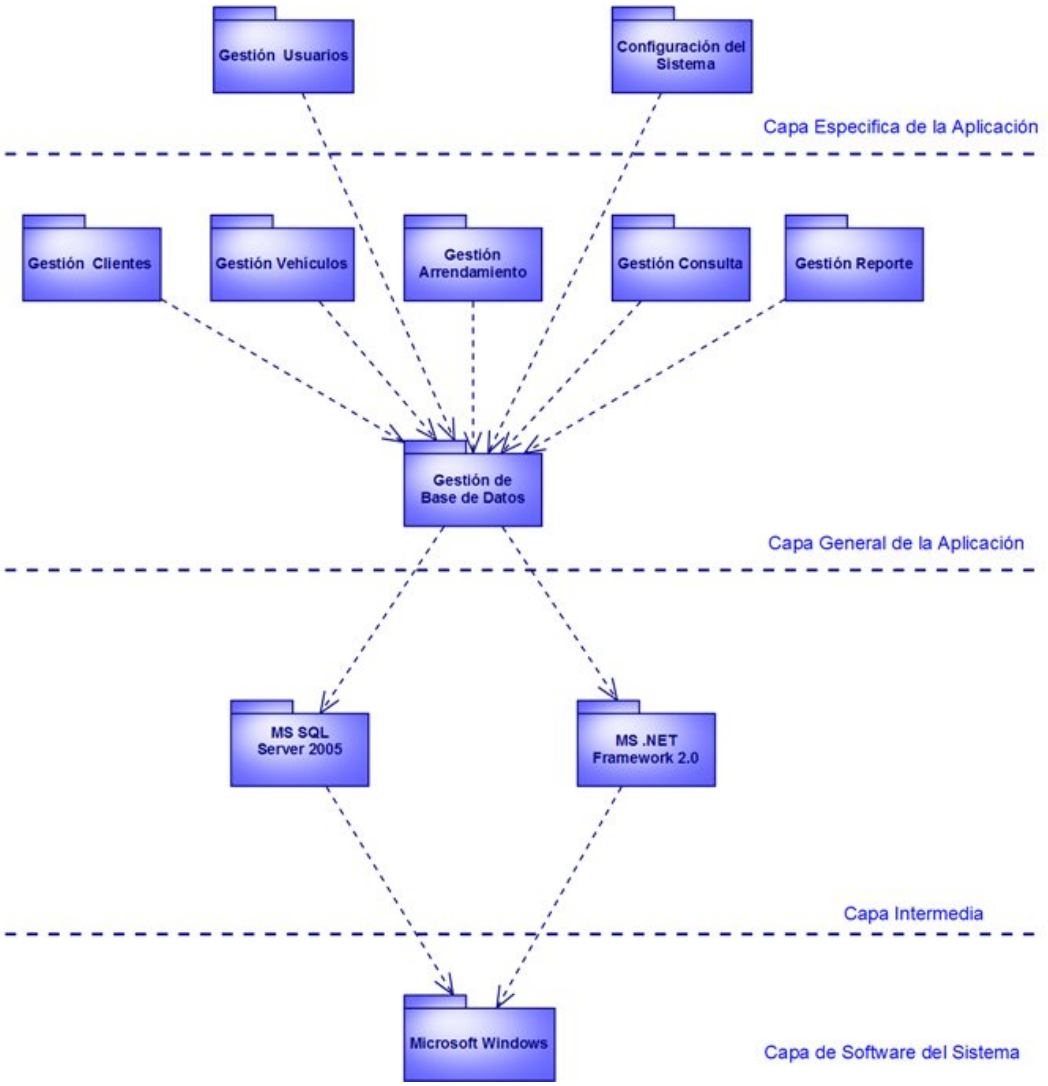


Figura 4.2 Diagrama de Capas del Sistema.

4.4 Diseño de la Base de Datos

El diseño de la base de datos es un elemento fundamental durante el desarrollo de cualquier proyecto de software, ya que de esta manera se pueden evaluar el esquema de almacenamiento de la información antes de implementarla, esta etapa es muy importante ya que ayuda a garantizar en

todo momento la integridad de los datos y evitar errores tales como lo son la conexión errada de las tablas y la duplicidad de la información.

Cada una de las tablas identificadas y que se describen están vinculadas entre sí por alguno de sus atributos. A esta relación se le llama dependencia funcional.

Las dependencias identificadas en el diagrama (ver Figura 4.3) corresponden todas a claves foráneas o externas incluidas en la base de datos del sistema “SAGEAV”. En el diagrama, las relaciones no apuntan directamente a los atributos que guardan una dependencia funcional entre sí. La pequeña llave al final de la línea de la relación indica que la clave principal de esa tabla determina funcionalmente a la clave foránea del mismo nombre de la tabla al extremo contrario de la relación.

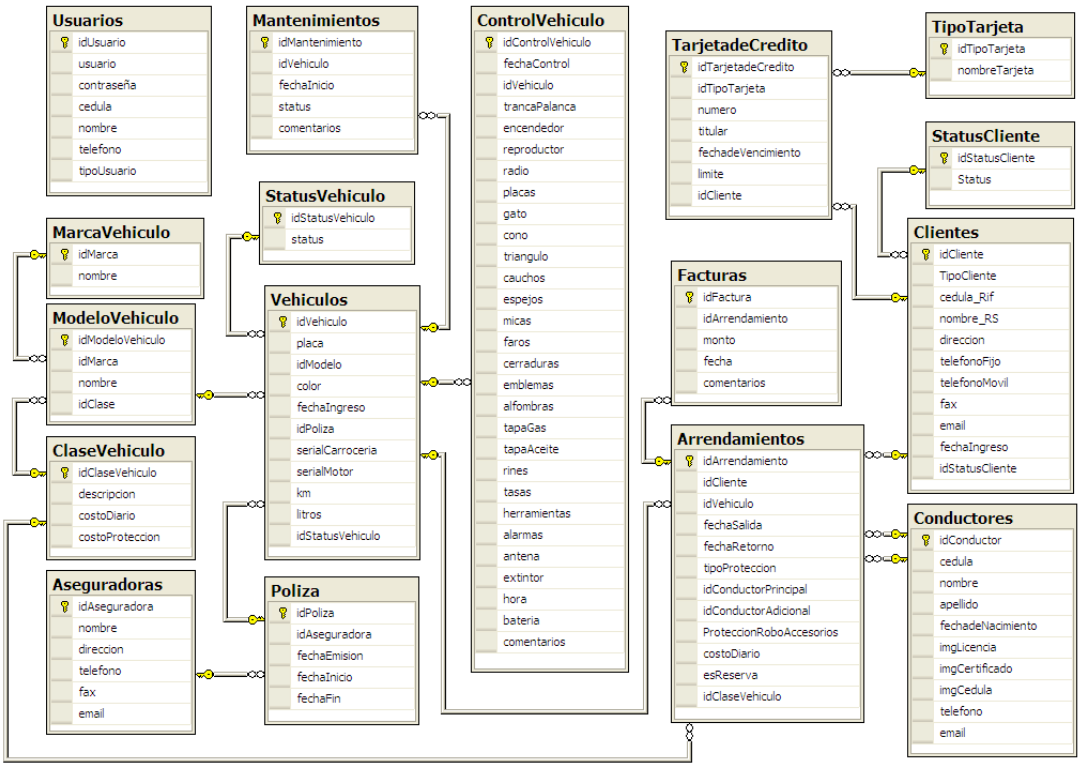


Figura 4.3 Modelo de la Base de Datos SAGEAV

4.4.1 Descripción de la base de datos

La base de datos del sistema SAGEAV está formada por un conjunto de tablas que serán descritas a continuación.

- **Tabla Clientes:** Almacena los datos principales del Cliente.

Tabla 4.1 Descripción de la tabla Cliente

Nombre	Tipo
idCliente	int
tipoCliente	int
cedula_Rif	varchar(10)
Nombre_Rs	varchar(50)
direccion	text
telefonoFijo	char(11)
telefonoMovil	char(11)
fax	char(11)
email	varchar(50)
fechaIngreso	datetime
idStatusCliente	int

- **Tabla TarjetadeCredito:** Se utiliza para registrar una o más tarjetas de crédito a cada cliente.

Tabla 4.2 Descripción de la tabla Tarjeta de Crédito

Nombre	Tipo
idTarjetadeCredito	int
idTipoTarjeta	int
numero	char(20)
titular	varchar(50)

fechadeVencimiento	datetime
limite	numeric(6,2)
idCliente	int

- **Tabla Vehículos:** Almacena la información de los vehículos en inventario.

Tabla 4.3 Descripción de la tabla vehículos

Nombre	Tipo
idVehiculo	int
placa	char(6)
idModelo	int
color	varchar(20)
fechaIngreso	datetime
idPoliza	int
serialCarroceria	varchar(50)
serialMotor	varchar(50)
km	numeric(6,2)
litros	numeric(6,2)
idStatusVehiculo	int

- **Tabla ModeloVehÍculo:** Especifica el modelo del Vehículo.

Tabla 4.4 Descripción de la tabla Modelo del Vehículo

Nombre	Tipo
idModeloVehiculo	int
idMarca	int
nombre	varchar(50)
idClase	int

- **Tabla ControlVehÍculo:** Necesaria para llevar el control de las características de cada vehículo al momento del retorno al garaje.

Tabla 4.5 Descripción de ControlVehículo

Nombre	Tipo
idControlVehiculo	int
idVehiculo	int
fechaControl	datetime
trancaPalanca	bit
encendedor	bit
reproductor	bit
radio	bit
placas	bit
gato	bit
cono	bit
triangulo	bit

cauchos	bit
espejos	bit
micas	bit
faros	bit
cerraduras	bit
emblemas	bit
alfombras	bit
tapaGas	bit
tapaAceite	bit
rines	bit
tasas	bit
herramientas	bit
alarmas	bit
antena	bit
extintor	bit
hora	bit
bateria	bit
Comentarios	text

- **Tabla Arrendamiento:** Registra la información necesaria correspondiente a cada arrendamiento.

Tabla 4.6 Descripción Arrendamiento

Nombre	Tipo
idArrendamieto	int
idCliente	int
idVehiculo	int
fechaSalida	datetime
fechaRetorno	datetime
tipoProteccion	int
idConductorPrincipal	int
idConductorAdicional	int
proteccionRoboAccesorios	bit
costoDiario	numeric(6,2)
esReserva	bit
idClaseVehiculo	int

- **Tabla Facturas:** Registra la información de las facturas correspondientes a cada pago asociada a un arrendamiento.

Tabla 4.7 Descripción de la tabla Facturas

Nombre	Tipo
idFacturas	int
idArrendamiento	int
monto	decimal(6,2)
fecha	datetime
comentarios	text

- **Tabla Usuarios:** Registra los datos de cada usuario del sistema.

Tabla 4.8 Descripción de la tabla Usuarios

Nombre	Tipo
idUsuario	int
usuario	varchar(50)
Contraseña	varchar(50)
cedula	varchar(10)
nombre	varchar(50)
telefono	char(11)
tipoUsuario	int

- **Tabla Mantenimientos:** Información que justifica los mantenimientos realizados a un vehículo.

Tabla 4.9 Descripción de la tabla Mantenimientos

Nombre	Tipo
idMantenimiento	int
idVehiculo	int
fechaInicio	datetime
status	int
comentarios	text

- **Tabla TipoTarjeta:** Almacena los distintos tipos de tarjetas de crédito en el sistema.

Tabla 4.10 Descripción de la tabla TipoTarjeta

Nombre	Tipo
idTipoTarjeta	int
nombreTarjeta	varchar(50)

- **Tabla StatusCliente:** Registra el estado en que se encuentra un cliente actualmente.

Tabla 4.11 Descripción de la tabla StatusCliente

Nombre	Tipo
idStatusCliente	int
status	varchar(50)

- **Tabla Conductores:** Registra los datos de cada conductor asociado a un arrendamiento, incluyendo las imágenes de sus documentos.

Tabla 4.12 Descripción de la tabla Conductores

Nombre	Tipo
idConductor	int
cedula	varchar(10)
nombre	varchar(50)
apellido	varchar(50)

fechadeNacimiento	datetime
imgLicencia	image
imgCertificado	image
imgCedula	image
telefono	char(11)
email	varchar(50)

- **Tabla Polizas:** Registra los datos de póliza seguros correspondientes a cada vehículo.

Tabla 4.13 Descripción de la tabla Polizas

Nombre	Tipo
idPoliza	int
idAseguradora	int
fechaEmision	datetime
fechaInicio	datetime
fechaFin	datetime

- **Tabla StatusVehÍculos:** Almacena los distintos status en los cuales puede estar un vehículo.

Tabla 4.14 Descripción de la tabla StatusVehiculos

Nombre	Tipo
idStatusvehiculo	int
status	varchar(50)

- **Tabla MarcaVehÍculo:** Representa la lista de marcas de vehículos.

Tabla 4.15 Descripción de la tabla MarcaVehículo

Nombre	Tipo
idMarca	int
nombre	varchar(50)

- **Tabla ClaseVehÍculo:** Contiene las distintas clases de vehículos por las cuales se distinguen los precios de arrendamiento.

Tabla 4.16 Descripción de la tabla ClaseVehículo

Nombre	Tipo
idClaseVehiculo	int
descripcion	varchar(50)
costoDiario	decimal(6,2)
costoProteccion	decimal(6,2)

- **Tabla Aseguradoras:** Datos de contactos referentes a las agencias de seguro de las pólizas de vehículos.

Tabla 4.17 Descripción de la tabla Aseguradoras

Nombre	Tipo
idAseguradora	int
nombre	varchar(50)
dirección	text

telefono	char(11)
fax	char(11)
email	varchar(50)

4.5 Diseño de la Interfaz

Uno de los aspectos más importantes del proceso de desarrollo de software es el diseño de la interfaz de usuario, su importancia radica en que, por medio de esta se hace posible la interacción entre el usuario y el sistema.

Se deben consideran aspectos asociados a facilitar dicha interacción, tales como: realizar un diseño de pantallas que cumplan con las siguientes normas; deben ser claras, sencillas, no se deben recargar de información, las pantallas deben ser consistentes, se deben usar los colores adecuados y con una buena selección de menús de usuarios.

Las interfaces del sistema SAGEAV fueron desarrolladas usando como herramienta Visual Basic 2005.

4.5.1 Acceso al sistema

Al iniciar el sistema SAGEAV se presenta una ventana de validación de usuario (Figura 4.4) con la cual se permite ingresar el correspondiente nombre de usuario y contraseña, si estos datos corresponden a un usuario existente en el sistema se permite su acceso con las restricciones definidas para el mismo.

Figura 4.4 Ventana de validación de usuario

4.5.2 Ventana Principal

Al iniciar el sistema se muestra la ventana principal (ver Figura 4.5) con la cual se efectuará la mayoría de las interacciones con el usuario a través del menú principal, además de contar con dos de las consultas más importantes para el día a día, que son “Vehículos por Entregar” y “Reservas para Hoy”.

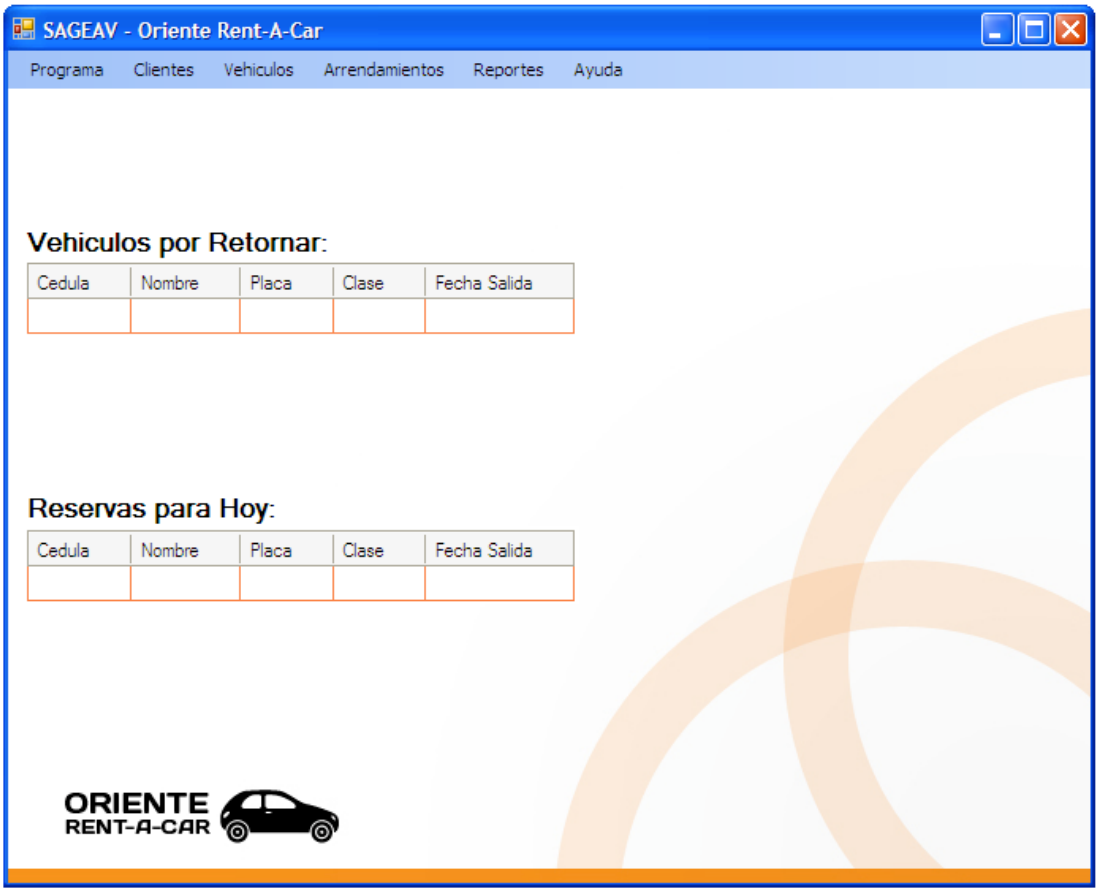


Figura 4.5 Ventana principal del sistema

4.5.3 Menú principal

El menú principal (ver Figura 4.6) proporciona una interfaz de fácil acceso a cada una de las funcionalidades del sistema.

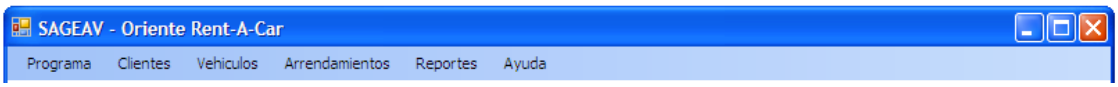


Figura 4.6 Menú principal

4.5.3.1 Menú “Programa”

Este menú (ver Figura 4.7) proporciona las operaciones básicas referidas a los usuarios y a la configuración del sistema, así como la opción de salir del sistema.

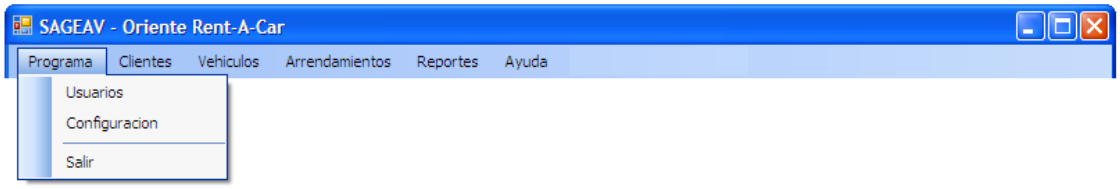


Figura 4.7 Menú “Programa”

4.5.3.1.1 Ventana “Usuarios”

Esta ventana (ver Figura 4.8) permite llevar a cabo las operaciones esenciales referidas a la administración de usuarios.

Figura 4.8 Ventana "Usuarios"

4.5.3.1.2 Ventana "Configuración"

Este menú (ver Figura 4.9) proporciona las operaciones básicas referidas a la configuración del sistema.

Figura 4.9 Ventana "Configuración

4.5.3.2 Menú "Clientes"

El presente menú (ver Figura 4.10) permite al usuario manejar los datos de los clientes a través de las siguientes funciones: Nuevo y Consultas.

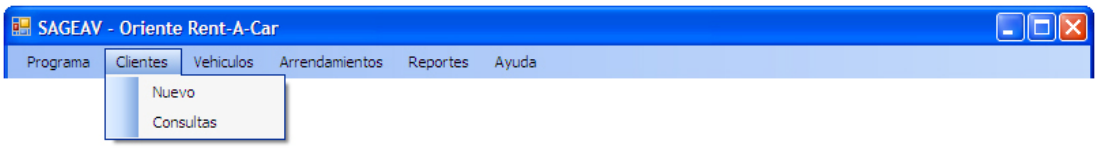


Figura 4.10 Menú "Clientes"

4.5.3.2.1 Ventana "Nuevo"

En la ventana Agregar Cliente (ver Figura 4.11) se permite al usuario agregar los datos de los clientes.

Figura 4.11 Ventana "Nuevo"

4.5.3.2.2 Ventana "Consultas"

En esta ventana (ver Figura 4.12) se pueden consultar los datos correspondientes a los clientes existentes en el sistema, llevar a cabo operaciones en estos datos, como lo son modificar y eliminar registros.

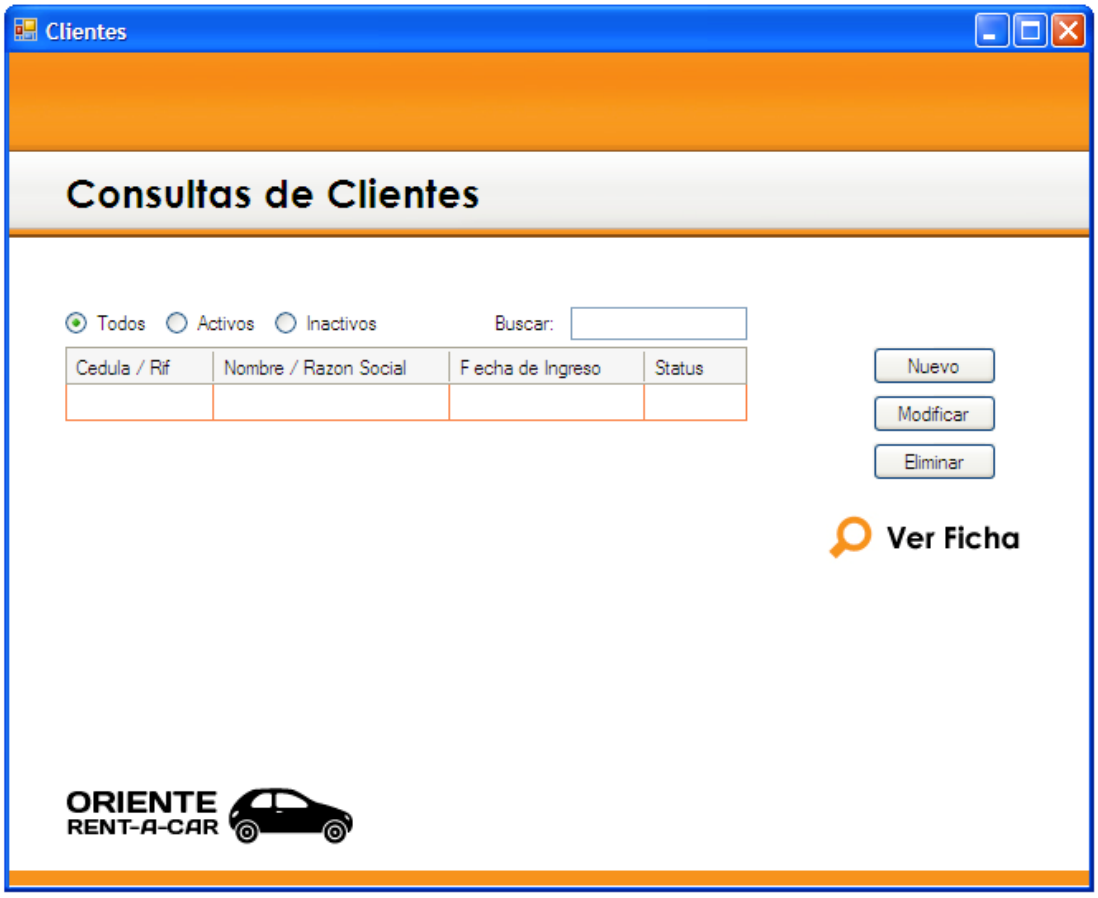


Figura 4.12 Ventana "Consultas"

4.5.3.3 Menú "Vehículos"

Este menú (ver Figura 4.13) le proporciona al usuario la facilidad de agregar, consultar los datos de cualquier vehículo en la empresa, así como llevar el control de los mantenimientos asignados a cada auto.

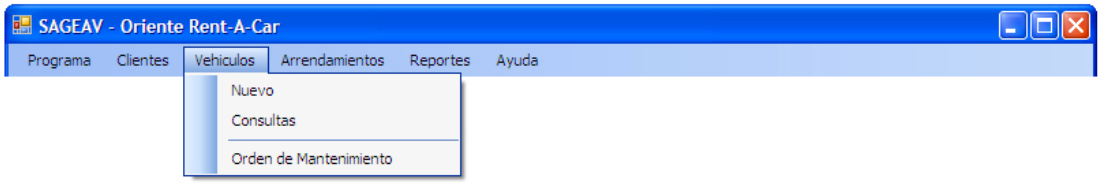


Figura 4.13 Menú "Vehículos"

4.5.3.3.1 Ventana "Nuevo"

En la ventana Agregar Vehículo (ver Figura 4.14) se permite al usuario agregar los datos de los nuevos vehículos.

Figura 4.14 Ventana "Nuevo"

4.5.3.3.2 Ventana “Consultas”

En esta ventana (ver Figura 4.15) se pueden consultar los datos correspondientes a los vehículos existentes en el sistema, llevar a cabo operaciones en estos datos, como lo son modificar y eliminar registros.

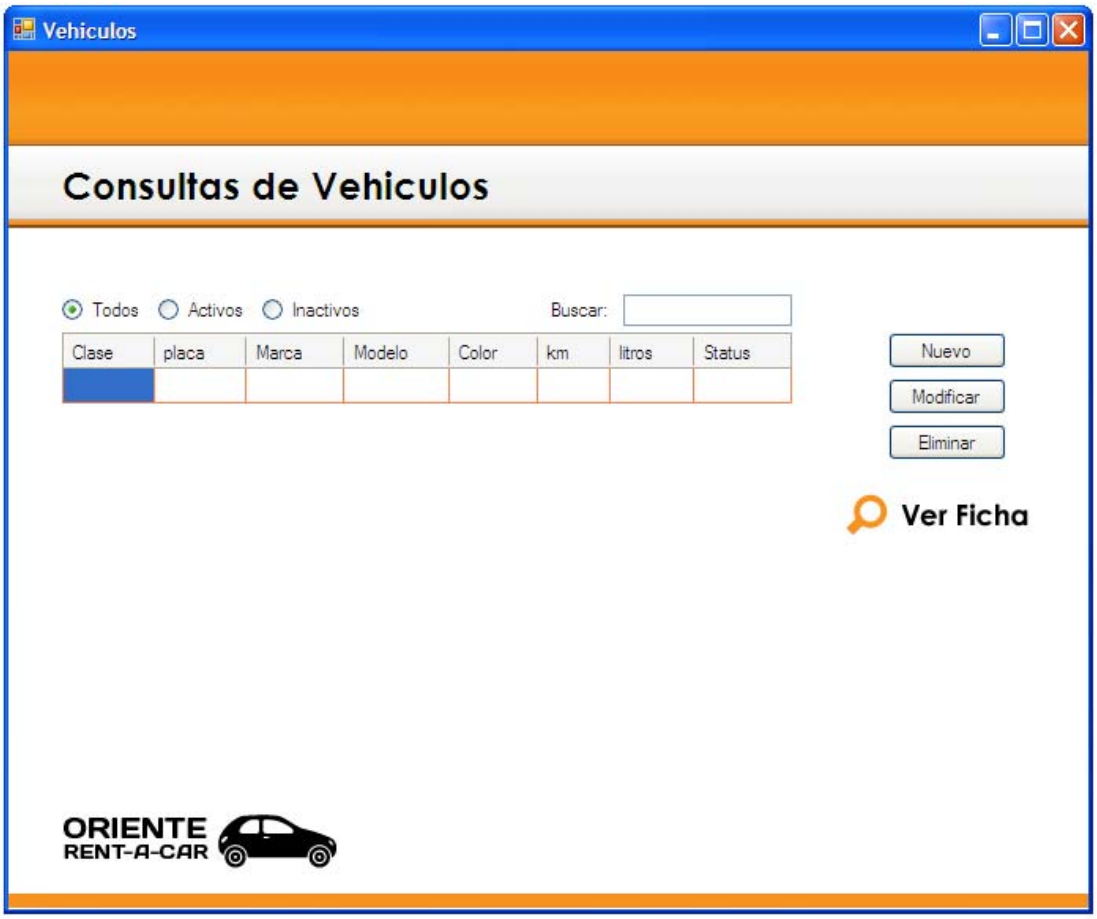


Figura 4.15 Ventana “Consultas”

4.5.3.3.3 Ventana “Orden de Mantenimiento”

En esta ventana (ver Figura 4.16) se permite al usuario ingresar los datos correspondientes a los mantenimientos asignados a un vehículo especifico.

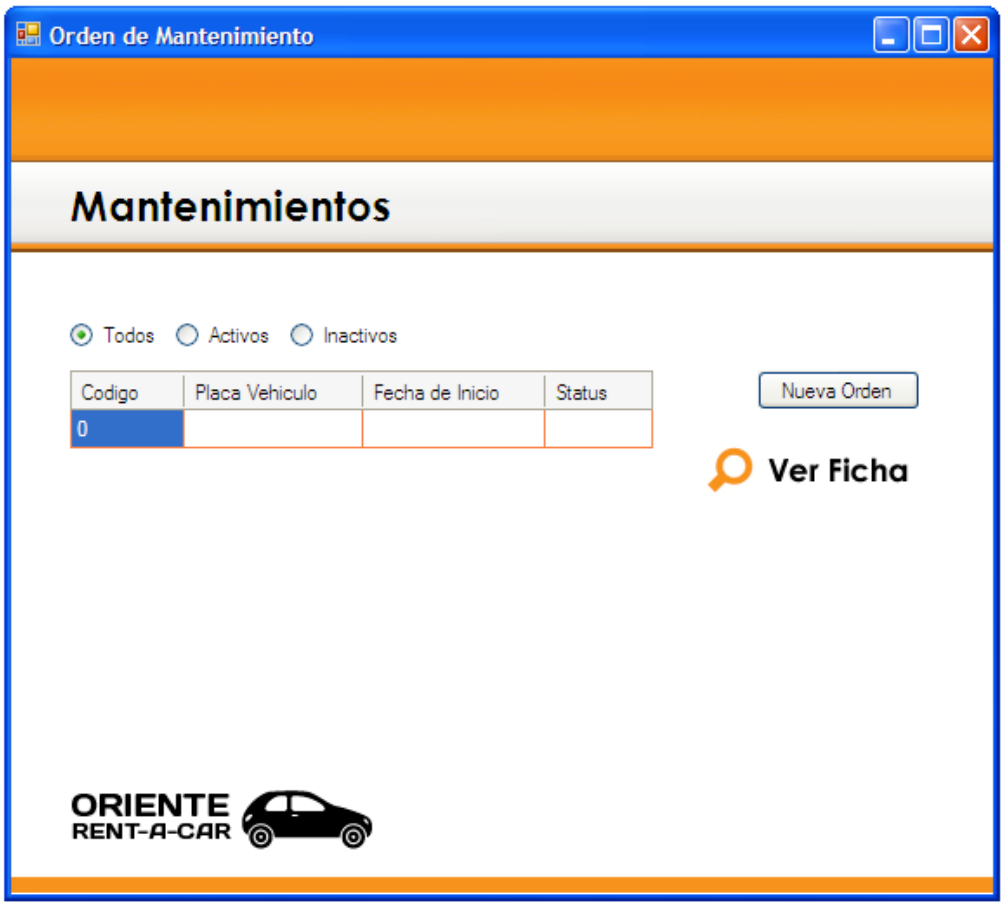


Figura 4.16 Ventana "Orden de Mantenimiento"

4.5.3.4 Menú "Arrendamiento"

El menú Arrendamiento (ver Figura 4.17) le proporciona al usuario la opción de reservar, alquilar, recibir vehículos y consultar arrendamientos.

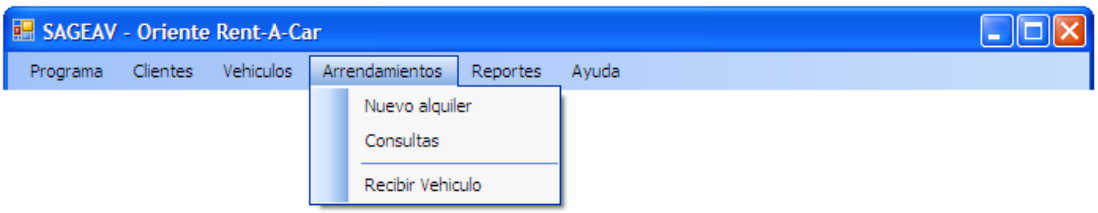


Figura 4.17 Menú "Arrendamientos"

4.5.3.4.1 Ventana "Nuevo Alquiler"

En esta ventana (ver Figura 4.18) ingresan los datos correspondientes a un arrendamiento, donde se le asigna un vehículo al cliente que lo alquila, así como las opciones del arrendamiento, que son el tipo de seguro, los conductores, y la fecha de salida del vehículo de la agencia y de retorno del mismo.

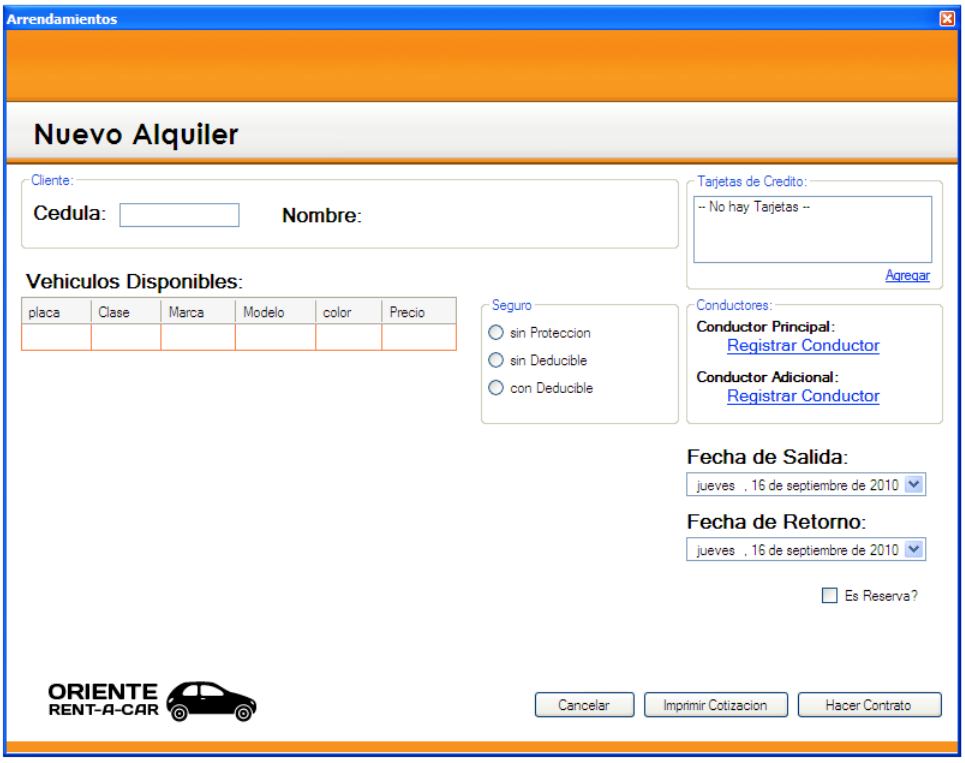


Figura 4.18 Ventana "Nuevo Alquiler"

4.5.3.4.2 Ventana “Consultas”

En esta Ventana menú (ver Figura 4.19) se realizan las consultas de los arrendamientos que se están llevando a cabo, así como los que se mantienen en el historial del sistema.

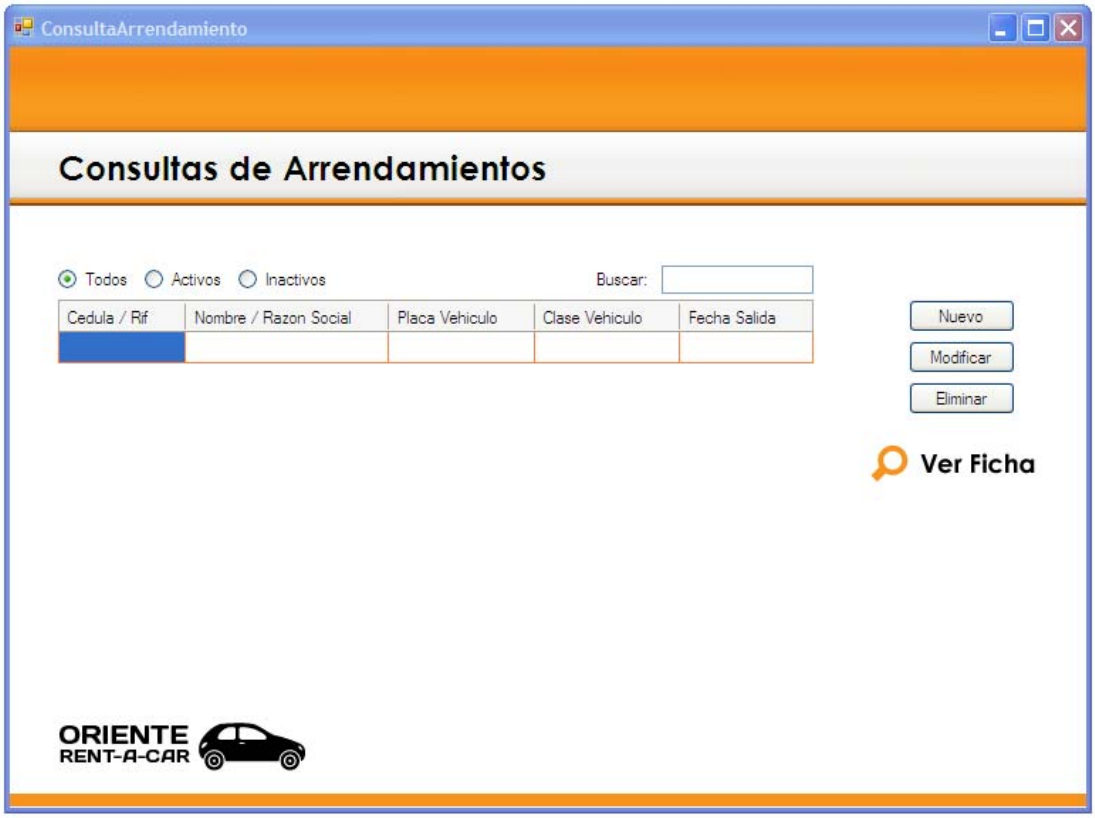


Figura 4.19 Ventana “Consultas”

4.5.3.4.3 Ventana “Recibir Vehículo”

A través de la ventana siguiente (ver Figura 4.20) se realizan las operaciones de retorno del vehículo por parte del cliente. Se lleva a cabo el chequeo de control y se cierra el contrato.

Figura 4.20 Ventana "Recibir vehículo"

4.5.3.4 Menú "Reportes"

Este menú (ver Figura 4.21) genera diversos tipos de reportes al usuario, tales como: de clientes, de vehículos, de arrendamientos, de Mantenimientos y de facturación.

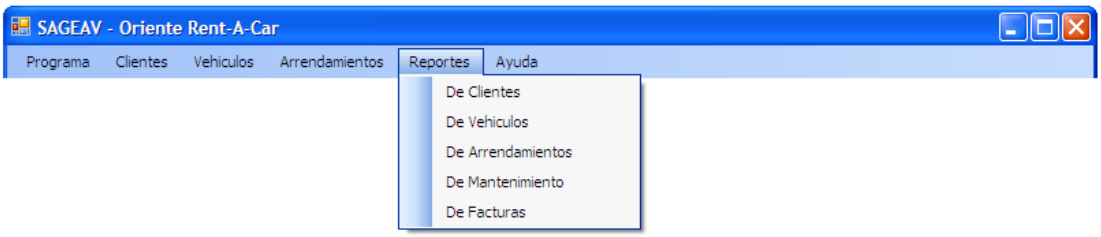


Figura 4.21 Menú "Reportes"

4.5.3.6 Menú "Ayuda"

Este menú (ver Figura 4.22) proporciona las herramientas necesarias para el desempeño del software al usuario a través del manual de usuario.

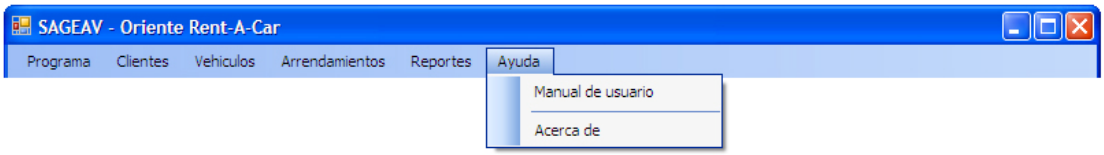


Figura 4.22 Menú "Ayuda"

4.6 Diagrama de Secuencia

El diagrama de secuencia se encarga de mostrar las interacciones necesarias entre los objetos y los actores del sistema para llevar a cabo la funcionalidad de los casos de uso. El mismo contendrá las instancias de los actores, los objetos de diseño y las transmisiones de mensajes entre éstos, ordenadas según el tiempo en que tienen lugar. A continuación presentaremos el diagrama de secuencia del caso de uso "Gestionar Arrendamiento" (ver Figura 4.23).

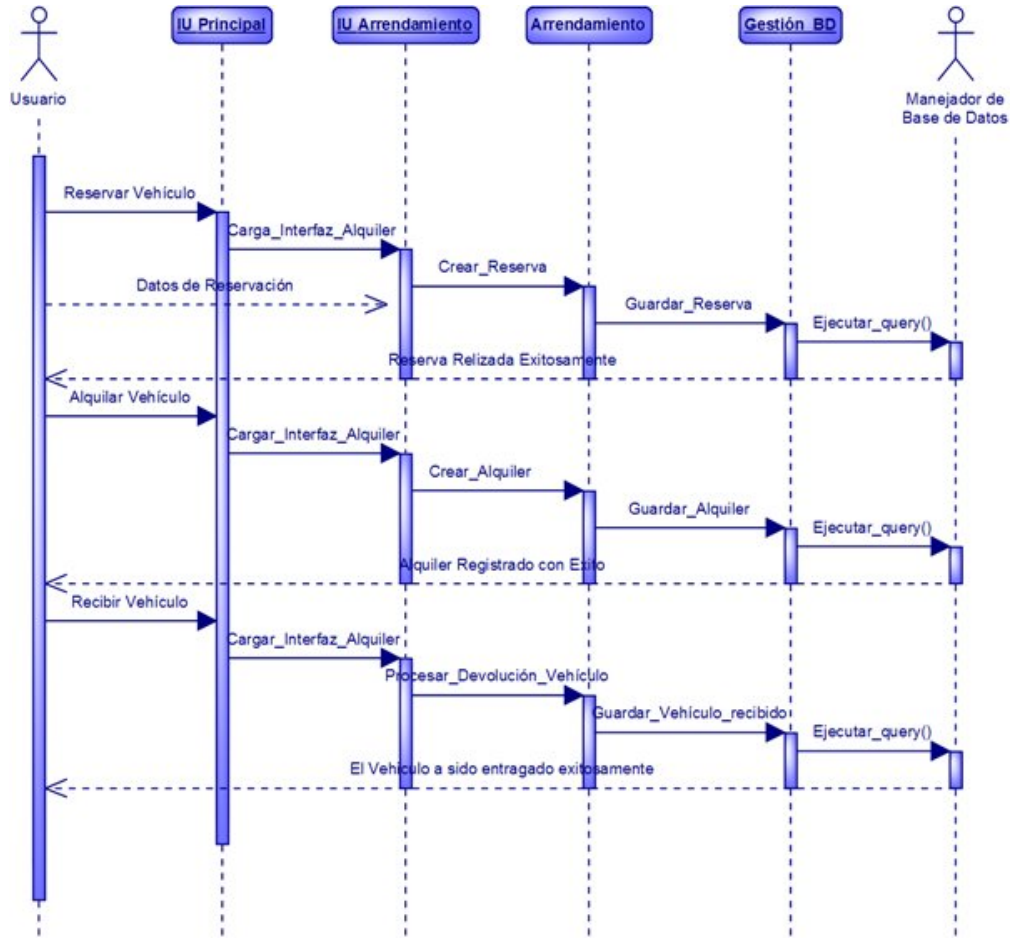


Figura 4.23 Diagrama de secuencia del caso de uso "Gestionar Arrendamiento"

4.7 Diagrama de Componentes

El diagrama de componentes se encarga de mostrar el modelado de la estructura del software, incluyendo las dependencias entre los componentes de software. Cada componente es un módulo de software y hace parte de la vista física del sistema, el diagrama de componentes del sistema lo podemos apreciar en la Figura 4.24.

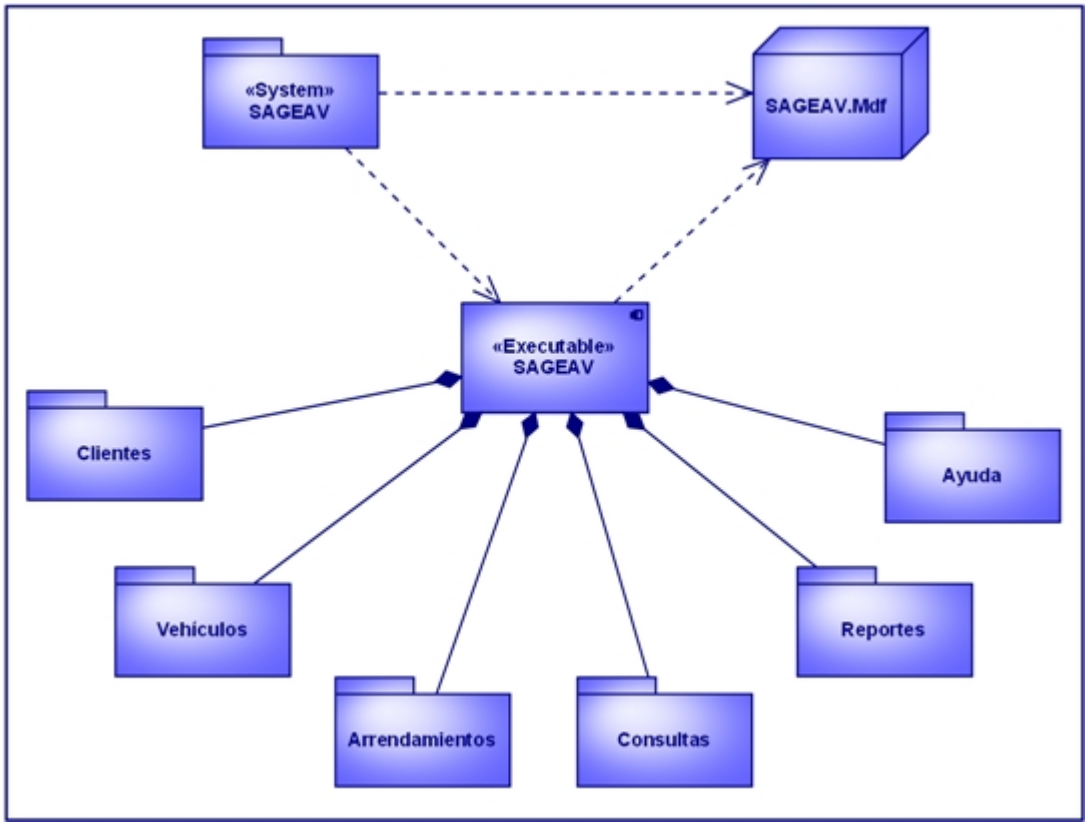


Figura 4.24 Diagrama de Componentes

4.8 Resumen de la Fase de Elaboración

Durante la presente fase en el desarrollo de este proyecto se logro concretar de manera significativa la estructura del software a implementar, además de definir el diseño de estructuras básicas como lo son el diagrama de clases, modelo de la base de datos y las interfaces de usuarios, con esto logramos alcanzar una visión bastante aproximada de la estructura final del sistema.

CAPÍTULO 5: FASE DE CONSTRUCCIÓN

Esta tercera fase del Proceso Unificado de Desarrollo del Software, tiene como propósito la obtención de un producto en su versión operativa inicial, el cual deberá tener la calidad adecuada para su aplicación y cumplir con todos los requisitos establecidos en las fases anteriores.

Luego de cumplir con los requisitos y análisis en las fases anteriores, se realizará, para esta fase, implementaciones y pruebas al Sistema de SAGEAV, una vez completado el diseño de su arquitectura, es decir, al programar y probar los subsistemas desarrollados se realizarán integraciones hasta obtener el sistema completo.

Es obvio que la arquitectura base del sistema está ya diseñada totalmente, y una buena parte de ésta es ejecutable. En líneas generales, en esta fase lo que resta es aplicar el flujo de trabajo que corresponde según la naturaleza de la fase y la línea base de la arquitectura lograda para el sistema. Esta fase de construcción demanda la aplicación del flujo de trabajo en todas sus instancias, pero el esfuerzo se concentra fundamentalmente en las actividades de implementación y prueba, las otras actividades por lo ya logrado en el sistema no requieren de mayor empleo.

5.1 Herramientas Utilizadas

Para el diseño de la interfaz del Sistema de SAGEAV se usa la herramienta de programación **Visual Basic 2005**, la cual se ejecuta sobre plataforma Windows. Además se utiliza la base de datos Microsoft Access para la implementación de las mismas. Tanto el lenguaje como el administrador de la

base de datos, se establecieron de acuerdo a las herramientas disponibles y con las cuales se contaba con licencia aprobada por la empresa.

A continuación una breve descripción:

- **WINDOWS:** es un sistema operativo gráfico para computadoras personales cuyo propietario es la empresa Microsoft. La primera versión popular, Windows 3.1 es una interfaz gráfica que funciona en MS-DOS. Éste es el sistema operativo más usado en el mundo.

- **SQL SERVER 2005:** es una base de datos visual. Access contiene herramientas de diseño y programación reservadas a los usuarios con mayor experiencia, aunque incluye bases de datos listas para ser usadas. Éstas están preparadas para tareas muy comunes que cualquiera puede realizar es un momento determinado.

- **VISUAL BASIC 2005:** es un lenguaje de programación desarrollado por Alna Cooper para Microsoft. El lenguaje de programación es un dialecto de BASIC con importantes añadidos. Su primera versión fue presentada en 1991 con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilita la creación de interfaces gráficas y en cierta medida también la programación misma.

  Es un lenguaje de fácil aprendizaje pensado tanto para programadores principiantes como expertos, guiado por eventos, y centrado en un motor de formularios que facilita el rápido desarrollo de aplicaciones gráficas. Su sintaxis, derivada del antiguo BASIC,

ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. Se ha agregado una implementación limitada de la programación orientada a objetos (los propios formularios y controles son objetos), aunque sí admite el polimorfismo mediante el uso de los Interfaces, no admite la herencia. No requiere de manejo de punteros y posee un manejo muy sencillo de cadenas de caracteres. Posee varias bibliotecas para manejo de bases de datos, pudiendo conectar con cualquier base de datos a través de ODBC (Informix, DBase, Access, MySQL, SQL Server, PostgreSQL, etc.) a través de ADO.

Es utilizado principalmente para aplicaciones de gestión de empresas, debido a la rapidez con la que puede hacerse un programa que utilice una base de datos sencilla, además de la abundancia de programadores en este lenguaje.

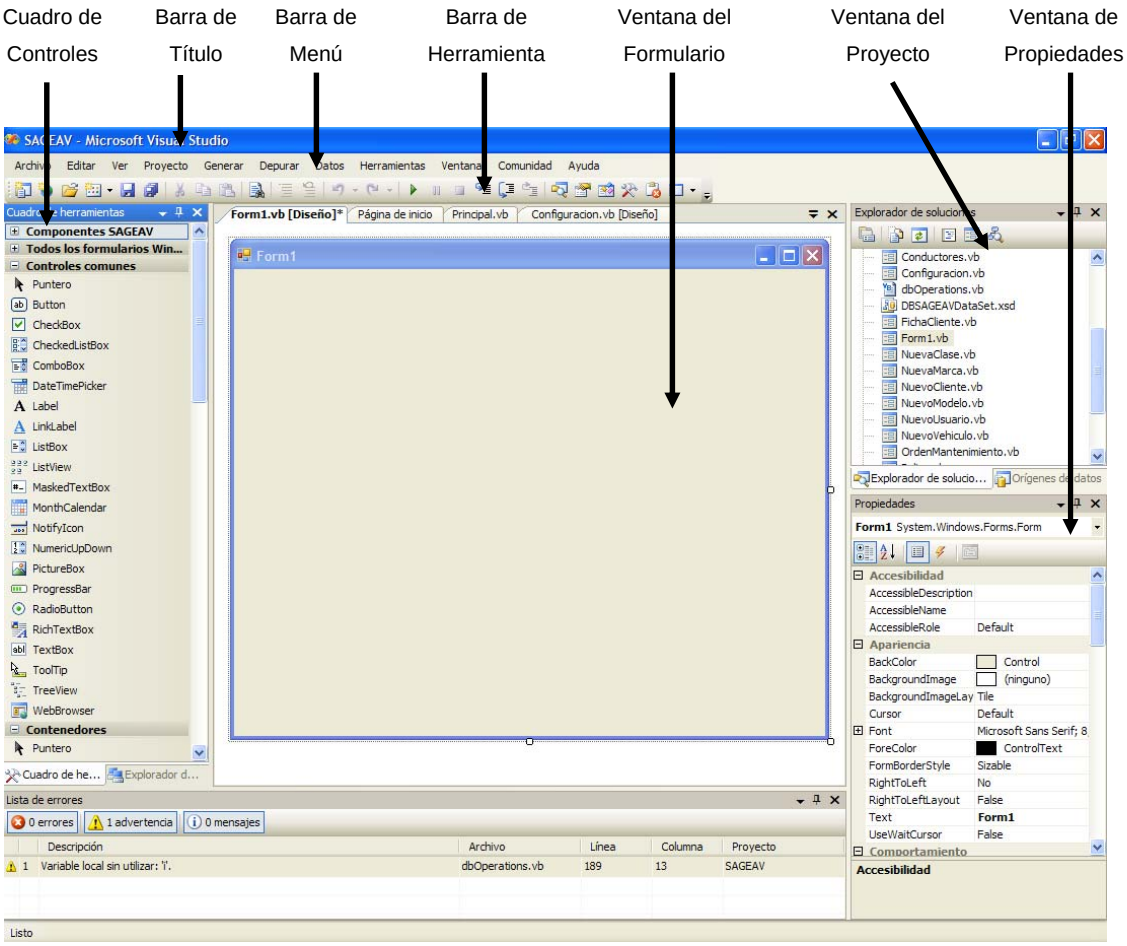
El compilador de Microsoft genera ejecutables que requieren una DLL para que funcionen, en algunos casos llamada MSVBVMxy.DLL (acrónimo de "Microsoft Visual Basic Virtual Machine x.y", siendo x.y la versión) y en otros VBRUNXXX.DLL ("Visual Basic Runtime X.XX"), que provee todas las funciones implementadas en el lenguaje. Además existen un gran número de bibliotecas (DLL) que facilitan el acceso a muchas funciones del sistema operativo y la integración con otras aplicaciones. Sin embargo esto sólo es una limitación en sistemas obsoletos, ya que las bibliotecas necesarias para ejecutar programas en Visual Basic vienen de serie en todas las versiones de Windows desde Windows 2000.

Es un lenguaje de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla. Es también un programa basado en objeto.

5.1.1 Entorno de Visual Basic

El entorno de desarrollo de Visual Basic es de naturaleza sencilla y flexible pero dotada de un gran número de componentes prefabricados que facilitan notablemente el desarrollo de programas.

Cuando se inicia Visual Basic 2005., aparece en la pantalla una configuración similar a la mostrada en la Figura 5.1. En ella se pueden distinguir los siguientes elementos:



Fuente: Microsoft Visual Basic .net 2005.

Figura 5.1 Aspecto Inicial de Entorno de Visual Basic

En la parte superior de la ventana principal, se encuentra la barra de título, la barra de menú y la barra de herramientas, el cuadro de controles se dispone en la parte izquierda de la pantalla, ésta contiene los controles básicos para desarrollar aplicaciones. En la parte derecha de la ventana principal se disponen dos ventanas verticales; la superior es la ventana de proyectos, que muestra los formularios y otros módulos de programas que forman parte de la aplicación y la ventana inferior es la de propiedades, en la que se pueden ver y manipular las propiedades del objeto seleccionado.

Finalmente se tiene una ventana en la parte central de la ventana principal, denominada ventana de formulario.

5.1.2 La Ventana Principal de Visual Basic

Bajo la barra de título de esta ventana se encuentra el menú de Visual Basic, que se conocerá en la medida que se empleen sus opciones. En la parte inferior se tiene la barra de herramientas la cual mediante íconos puede recuperar un proyecto, guardar el actual, ejecutar el programa, entre otras opciones. Cada icono se corresponde con una opción de la barra de menú.

Finalmente, se tiene la caja de controles en la que se puede encontrar los componentes y controles básicos de un programa.

5.1.3 La Barra de Menú

Esta barra contiene las diferentes opciones del programa agrupadas en varias entidades principales, las cuales ofrecen los servicios más importantes.

5.1.4 La Barra de Herramientas

Debajo de la barra de menú se encuentra la barra de herramientas, un conjunto de botones que funcionan como atajos para ejecutar opciones y controlar el entorno de programación de Visual Basic.

5.1.5 El Cuadro de Controles

El cuadro de controles contiene todos los controles que se pueden insertar en la interfaz de usuario como: etiquetas, botones, cuadros de lista, barras de desplazamiento, menús, imágenes, formas geométricas, etc. Una vez que los controles han sido introducidos en el formulario se convierten en objetos, o elementos programables de la interfaz de usuario.

5.1.6 El Formulario

Todas las interfaces de las aplicaciones construidas en Visual Basic se fundamentan en los formularios (forms) en los que se pueden agregar los distintos controles encargados de interactuar con el usuario. Muchas aplicaciones se conforman de múltiples formularios.

5.1.7 La Ventana de Propiedades

La finalidad de esta ventana es facilitar la configuración de las propiedades del componente seleccionado. En la parte superior aparece el nombre del componente que está seleccionado, así como su tipo. La columna izquierda de la ventana muestra los nombres de las propiedades o eventos, mientras que la derecha contiene los valores que configuran su comportamiento.

5.1.8 La Ventana de Proyecto

Muestra los formularios, informes, tablas de la base de datos y otros módulos de programas que forman parte de la aplicación.

5.1.9 La Ventana de Código

El importante trabajo de codificación en Visual Basic está generalmente asociado a los métodos de respuesta a eventos. Para especificar el código necesario para ejecutar una acción en Visual Basic se debe hacer doble clic en el objeto o hacer clic en la caja de objeto de la ventana de código, y luego editar el código. También se pueden seleccionar otros procedimientos, cuya lista se muestra en la caja de procedimientos o eventos, para el objeto seleccionado (ver Figura 5.2).

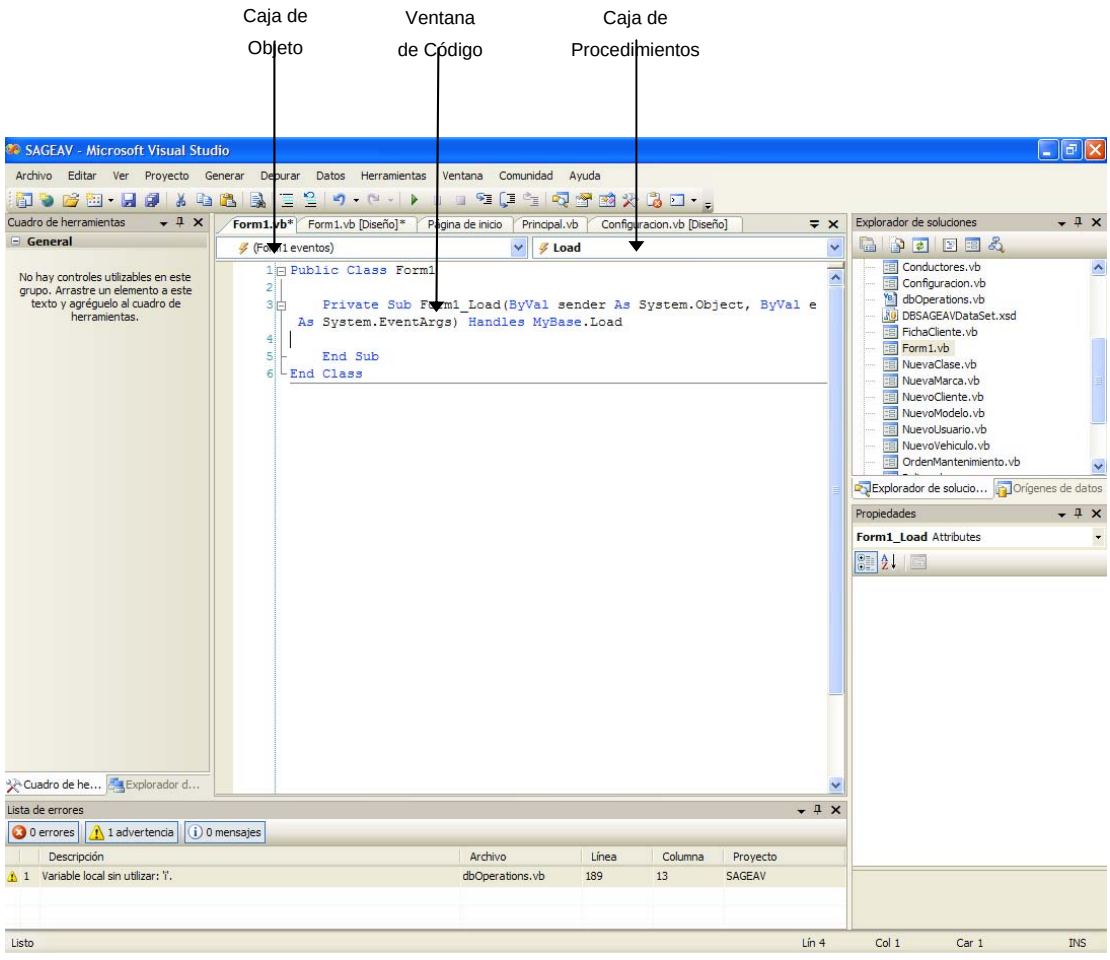


Figura 5.2 Ventana de Código de Visual Basic

5.2 Codificacion de las Estructuras de Datos

A continuación se muestra parte del código de tres de las clases más importantes del sistema, las cuales son **AccesoaSistema**, **NuevoCliente** y **NuevoVehículo**.

Public Class AccesoaSistema

```
   Private Sub btnEntrar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEntrar.Click
      Dim oCon As New dbOperations
      Dim htParam As New Hashtable
      Dim dt As New DataTable
      htParam.Add("@usuario", TextBox1.Text)
      htParam.Add("@contraseña", TextBox2.Text)
      oCon.Server = "gen\sqlnew"
      oCon.DataBase = "DBSAGEAV"
      oCon.UserId = "sa"
      oCon.Password = "sageav"
      dt = oCon.EjecutarProcedure("accesoaSistema", htParam)
      If oCon.errorEstatus Then
         If (Not dt Is Nothing) And (dt.Rows.Count > 0) Then
            Mensaje.Text = "Usuario registrado en la BD proceda    El Id es " +
dt.Rows(0)("idUsuario").ToString
            Me.Close()
            sageav.Visible = True
         Else
            Mensaje.Text = "No hay usuario registrado en la bd"
         End If
      Else
         Mensaje.Text = oCon.ErrorMsg
      End If
   End Sub

   Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged
      Dim oCon As New dbOperations
      Dim htParam As New Hashtable
      Dim dt As New DataTable
      htParam.Add("@usuario", TextBox1.Text)
      oCon.Server = "gen\sqlnew"
```

```
        oCon.DataBase = "DBSAGEAV"
        oCon.UserId = "sa"
        oCon.Password = "sageav"
        dt = oCon.EjecutarProcedure("verificarUsuario", htParam)
        If oCon.errorEstatus Then
            If (Not dt Is Nothing) And (dt.Rows.Count > 0) Then
                Mensaje.Text = "Usuario Valido " + dt.Rows(0)("usuario").ToString
            Else
                If (TextBox1.Text <> "") Then
                    Mensaje.Text = "invalido"
                End If
            End If
        Else
            Mensaje.Text = oCon.ErrorMsg
        End If
    End Sub
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        sageav.Close()
    End Sub
End Class
```

Public Class NuevoCliente

```
    Private Sub Clientes_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        natural.Checked = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Tarjetas.ShowDialog()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Conductores.ShowDialog()
    End Sub
```

```
    Private Sub RadioButton1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
natural.CheckedChanged
        Label2.Text = "Cedula:"
        Label3.Text = "Nombre:"
    End Sub

    Private Sub RadioButton2_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
juridico.CheckedChanged
        Label2.Text = "RIF:"
        Label3.Text = "Razon Social:"
    End Sub

    Private Sub cancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancelar.Click
        Me.Close()
    End Sub

    Private Sub guardar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles guardar.Click
        Dim oCon As New dbOperations
        Dim htParam As New Hashtable
        Dim dt As New DataTable
        Dim tipoCliente As Integer
        Dim Cedula As String
        Cedula = tipoCedula.Text + "-" + cedulaRif.Text
        If natural.Checked Then
            tipoCliente = natural.Tag
        Else
            tipoCliente = juridico.Tag
        End If
        htParam.Add("@tipoCliente", tipoCliente)
        htParam.Add("@cedula_Rif", Cedula)
        htParam.Add("@nombre_RS", nombre_RS.Text)
        htParam.Add("@direccion", direccion.Text)
        htParam.Add("@telefonoFijo", telefonoFijo.Text)
        htParam.Add("@telefonoMovil", telefonoMovil.Text)
        htParam.Add("@fax", fax.Text)
        htParam.Add("@email", email.Text)
        htParam.Add("@fechaIngreso", hoy.Value)
        oCon.Server = "gen\sqlnew"
        oCon.DataBase = "DBSAGEAV"
```

```
        oCon.UserId = "sa"
        oCon.Password = "sageav"
        dt = oCon.EjecutarProcedure("addCliente", htParam)
        If oCon.errorEstatus Then
            If (Not dt Is Nothing) And (dt.Rows.Count > 0) Then
                MsgBox("bien, otro?")
                Me.Close()
            Else
            End If
        Else
            MsgBox("algo esta mal: " + oCon.ErrorMsg)
        End If
    End Sub
End Class
```

Public Class NuevoVehiculo

```
    Private Sub Vehiculo_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'TODO: esta línea de código carga datos en la tabla
'DBSAGEAVDataSet.Aseguradoras' Puede moverla o quitarla según sea
necesario.

Me.AseguradorasTableAdapter.Fill(Me.DBSAGEAVDataSet.Aseguradoras)
        'TODO: esta línea de código carga datos en la tabla
'DBSAGEAVDataSet.ClaseVehiculo' Puede moverla o quitarla según sea
necesario.

Me.ClaseVehiculoTableAdapter.Fill(Me.DBSAGEAVDataSet.ClaseVehiculo)
        'TODO: esta línea de código carga datos en la tabla
'DBSAGEAVDataSet.ModeloVehiculo' Puede moverla o quitarla según sea
necesario.
Me.ModeloVehiculoTableAdapter.Fill(Me.DBSAGEAVDataSet.ModeloVehicul
o)
    End Sub

    Private Sub NuevoVehiculoGuardar(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles confirmar.Click
        Dim poliza As String
        poliza = guardarPoliza()
        If poliza <> "" Then
            guardarVehiculo(poliza)
```

```
        Else
            MsgBox("no se pudo guardar")
        End If
    End Sub

    Private Sub cancelar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cancelar.Click
        Me.Close()
    End Sub

    Private Function guardarPoliza() As String
        Dim oCon As New dbOperations
        Dim oCon2 As New dbOperations
        Dim htParam As New Hashtable
        Dim htParam2 As New Hashtable
        Dim dt As New DataTable
        Dim dt2 As New DataTable
        Dim poliza As String
        htParam.Add("@idAseguradora", aseguradora.SelectedValue)
        htParam.Add("@numerodePoliza", numerodePoliza.Text)
        htparam2.Add("@numerodePoliza", numerodePoliza.Text)
        htParam.Add("@vence", vence.Value)
        oCon.Server = "gen\sqlnew"
        oCon.DataBase = "DBSAGEAV"
        oCon.UserId = "sa"
        oCon.Password = "sageav"
        oCon2.Server = "gen\sqlnew"
        oCon2.DataBase = "DBSAGEAV"
        oCon2.UserId = "sa"
        oCon2.Password = "sageav"
        dt = oCon.EjecutarProcedure("addPoliza", htParam)
        If oCon.errorEstatus Then
            If (Not dt Is Nothing) And (dt.Rows.Count > 0) Then
                MsgBox("bien, otro?")
            Else
            End If
        Else
            MsgBox("algo esta mal en la poliza: " + oCon.ErrorMsg)
        End If

        dt2 = oCon2.EjecutarProcedure("getPolizaId", htParam2)
        If oCon2.errorEstatus Then
            If (Not dt Is Nothing) And (dt2.Rows.Count > 0) Then
```

```
            MsgBox("bien, otro?")
        Else
        End If
    Else
        MsgBox("algo esta mal en la poliza: " + oCon2.ErrorMsg)
    End If
    poliza = dt2.Rows(0)("idPoliza").ToString
    Return poliza
End Function

Private Function guardarVehiculo(ByVal poliza As String) As Boolean
    Dim oCon As New dbOperations
    Dim htParamVehiculo As New Hashtable
    Dim dt As New DataTable
    htParamVehiculo.Add("@placa", placa.Text)
    htParamVehiculo.Add("@idModelo", modelo.SelectedValue)
    htParamVehiculo.Add("@color", color.Text)
    htParamVehiculo.Add("@fechaIngreso", hoy.Value)
    htParamVehiculo.Add("@idPoliza", poliza)
    htParamVehiculo.Add("@serialCarroceria", serialCarroceria.Text)
    htParamVehiculo.Add("@SerialMotor", serialMotor.Text)
    htParamVehiculo.Add("@km", 0)
    htParamVehiculo.Add("@litros", 0)
    htParamVehiculo.Add("@idStatusVehiculo", 1)
    oCon.Server = "gen\sqlnew"
    oCon.DataBase = "DBSAGEAV"
    oCon.UserId = "sa"
    oCon.Password = "sageav"
    dt = oCon.EjecutarProcedure("addVehiculo", htParamVehiculo)
    If oCon.errorEstatus Then
        If (Not dt Is Nothing) And (dt.Rows.Count > 0) Then
            MsgBox("bien, otro?")
        Else
        End If
    Else
        MsgBox("algo esta mal en el vehiculo: " + oCon.ErrorMsg)
    End If
End Function
```

5.3 Pruebas

La fase de prueba consiste básicamente en comprobar que el software realiza correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral. Entre los principales propósitos de las pruebas tenemos:

- Verificar la interacción entre los componentes.
- Verificar la integración apropiada de los módulos.
- Verificar que se satisfacen los requerimientos.
- Identificar los defectos y corregirlos antes de la instalación.

A continuación se describirán las pruebas de unidad y de integración aplicadas al sistema SAGEAV durante esta etapa de su desarrollo.

5.3.1 Pruebas de Unidad

La prueba de unidad se centra en el módulo. Usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo.

Al sistema SAGEAV se le realizaron las pruebas de caja negra, dichas pruebas verifican el comportamiento de la unidad observable externamente.

La prueba de la caja negra toma en cuenta la salida que el componente devolverá en función a una determinada entrada. Para la implementación de las mismas se determinaron las clases de equivalencia, las cuales

representan un conjunto de estados válidos o inválidos para diferentes entradas.

- Prueba de Unidad del Caso de uso “Gestionar Arrendamientos” (Tabla 5.1)

Tabla 5.1 Clases de equivalencia del caso de uso “Gestionar Cliente”

	Clase de Equivalencia	Válido	Inválido
1	Campo “Cedula” es Numérico	X	
2	Campo “Cedula” es Alfanumérico		X
3	Campo “Cedula” longitud es > 6 y < 10	X	
4	Campo “Cedula” longitud > 10		X
5	Campo “Cedula” longitud < 6		X
6	Campo “Teléfono Fijo” es Numérico	X	
7	Campo “Teléfono Fijo” es Alfanumérico		X
8	Campo “Teléfono Fijo” longitud > 11		X
9	Campo “Teléfono Fijo” longitud < 11		X
10	Campo “Teléfono Fijo” longitud = 11 Numérico	X	
11	Campo “Nombre” es Numérico	X	
12	Campo “Nombre” es Alfanumérico	X	
13	Campo “Nombre” longitud es <=0		X

- Prueba de Caja Negra del caso de uso “Generar Arrendamiento” (Tabla 5.2)

Tabla 5.2 Pruebas de caja negra del caso de uso “Gestionar Arrendamientos”

Caso de Prueba	Status	Clases Cubiertas
Nombre = “LT-210”	Válido	11,12
Nombre = “”	Inválido	13
Teléfono Fijo = “45”	Inválido	9
Teléfono Fijo = “G-210549KYTR”	Inválido	7
Cedula = 1150	Inválido	6
Cedula = 8754875	Válido	1,3
Cedula = ABC123	Invalido	2

5.3.2 Pruebas de Integración

A continuación se explica el proceso de integración del sistema utilizando como ejemplo **Acceso.vb**

Integrar datos de Acceso

Este caso de prueba verifica los datos del usuario que desea acceder al sistema.

Entrada: Nombre de Usuario y Contraseña.

Resultado: El usuario accede al sistema con los privilegios propios de su tipo.

Condiciones: Escoger una opción.

Procedimiento de Prueba:

1. El usuario ingresa su nombre de usuario.
2. El usuario ingresa su contraseña.
3. El usuario presiona el botón de acceder.

Si el nombre de usuario y la contraseña ingresada son validos, el usuario accede a la pantalla principal del sistema como se muestra en la Figura 5.3.

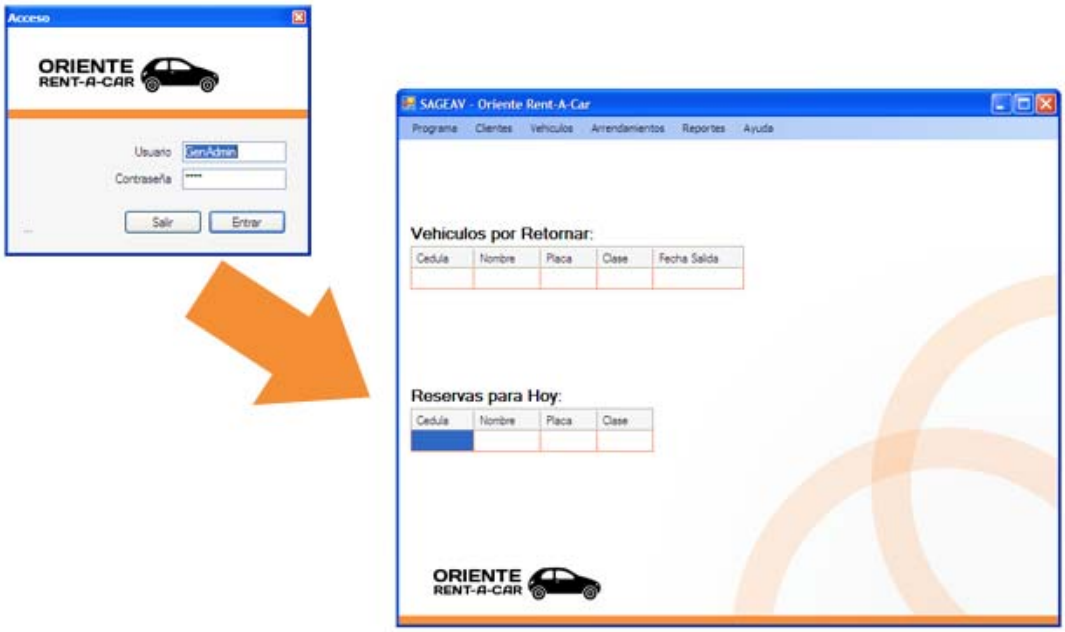


Figura 5.3 Ejemplo de prueba de Integración

CAPÍTULO 6: FASE DE TRANSICIÓN

6.1 Fase de Transición

La etapa de transición permite probar la calidad del software sobre la plataforma del entorno de operación. Es donde los usuarios empiezan a interactuar con el sistema y el desarrollador del software comienza a corregir defectos encontrados.

El objetivo principal en esta fase, es preparar la versión Beta del software e instalarlo en el entorno de trabajo para lo cual fue creado.

En las fases de inicio se establecieron requisitos que fueron analizados y diseñados en la fase de elaboración, para ser codificados en la etapa de construcción y finalmente implantado en la fase de transición. En esta fase se ha culminado el estudio del sistema y éste se encuentra operativo fundamentándose en el flujo de trabajo de pruebas con la finalidad de ajustarlo a los parámetros del entorno de trabajo del Departamento de Informática de empresas **Oriente Rent-A-Car** o alguna agencia arrendadora de vehículos de características similares**.**

6.2 Preparación de la Versión Beta

Se prepara documentos acerca del procedimiento de la aplicación, los cuales estarán dirigidos al personal del Departamento de Informática y de atención al cliente de la empresa **Oriente Rent-A-Car.**

Se seleccionan los usuarios Beta, es decir el Coordinador de Informática y los Analistas de Sistemas quienes poseen alta experiencia sobre el funcionamiento de los procesos que fueron automatizados.

6.3 Instalación de la Versión Beta

La instalación se desarrollo en el computador principal del Departamento de atención al cliente de la empresa **Oriente Rent-A-Car.** donde funcionara el proyecto.

Se realizo una presentación con las funciones principales del software con la finalidad de que el usuario tuviera las nociones necesarias para manejar sin ninguna dificultad la aplicación.

6.4 Reacción a los Resultados de las Pruebas

Se encontraron algunas fallas de codificación que fueron corregidas, no se encontró errores de mayor importancia que comprometieran la arquitectura del sistema desarrollado, por lo que no hubo necesidad de desarrollar ningún flujo de trabajo adicional.

6.5 Resumen de la Fase

En esta fase se instalo la versión Beta del software y se efectuaron las pruebas de aceptación con las cuales fueron localizados fallas menores en algunos componentes. Estos fallos fueron corregidos sin afectar la

arquitectura del sistema. De esta manera concluye la fase final del proceso de desarrollo de software con una versión completa y corregida del producto.

CONCLUSIONES

1) El empleo del Proceso Unificado de Desarrollo de Software conjuntamente con UML representa una poderosa herramienta para la construcción de un producto software, en virtud de que el primero garantiza la obtención de un software de calidad y el segundo plantea los modelos que dan representación abstracta al sistema.

2) Los modelos de análisis permitieron obtener de una forma más específica y precisa los requisitos del sistema y el análisis desarrollado permitió perfeccionar el modelo de funcionamiento.

3) UML mediante los diagramas que plantea, surge como un lenguaje determinante para el diseño conceptual de la estructura de software, ya que logra expresar de forma clara y flexible los aspectos funcionales del software a través de una comunicación directa e intuitiva por medio de representaciones.

4) Los diagramas de Casos de Uso del SAGEAV apoyaron la captura de requisitos potenciales proporcionando escenarios de interacción del sistema con los usuarios.

5) La fase de inicio estudia los requisitos necesarios para la implementación del SAGEAV, describiendo un plano del sistema como expresiones de lenguaje de programación, esquemas de bases de datos y componentes de software.

6) En la fase de elaboración se obtuvo la línea base de la arquitectura del SAGEAV desarrollada a través de los modelos de casos de uso, análisis y diseño, la construcción de las clases y subsistemas y la elaboración de la base de datos.

7) La base de datos fue diseñada mediante un modelo entidad relación, el cual permitió un fácil diseño de la misma, ya que éste permite una adecuada esquematización del conocimiento empleado por el sistema.

8) La interfaz de usuario fue de suma importancia debido a que permitió la comunicación del usuario con el SAGEAV y el intercambio de información.

9) La fase de construcción dio como resultado la obtención de la versión operativa inicial del SAGEAV, la cual se realizó cumpliendo los requisitos establecidos en las fases desarrolladas con anterioridad.

10) La realización de las pruebas fue de mucha importancia, ya que permitió la comprobación del código y el correcto funcionamiento, evaluando así la calidad del software y garantizando su eficiencia.

11) La elaboración del manual de usuario permitió una rápida familiarización del usuario con el SAGEAV.

12) La implementación del SAGEAV facilitó el control de todos los vehículos que tiene la empresa, bien sean alquilados, reservados o en mantenimiento, permitiendo un crecimiento laboral y personal de la misma.

13) SAGEAV le ha dado el avance tecnológico necesario para agilizar las actividades y poder brindar un mejor servicio, de allí que su implementación ha resuelto el problema planteado.

RECOMENDACIONES

1) Se recomienda la utilización del SAGEAV como soporte para llevar un rápido y confiable manejo del control de los vehículos arrendados, eliminando así el uso del sistema manual (tablas de Microsoft Excel), el cual genera un atraso operacional a la hora de realizar la búsqueda del material.

2) Elaborar una jornada de inventario periódica que ayude a la depuración y mantenimiento de la data almacenada en las bases de datos y así los usuarios poder contar con información actualizada y confiable.

3) Mantener suficiente espacio en disco duro, dado el posible incremento de la base de datos.

4) Consultar el manual de usuario al momento de surgir alguna duda con respecto al manejo del sistema.

5) Ingresar al día todas las listas de autos en reservas, alquiler e mantenimiento para mantener actualizado el sistema, y de esa manera llevar a cabo un control de inventario de una manera más eficiente.

6) Realizar planes de entrenamiento a los usuarios que faciliten la manipulación del SAGEAV.

BIBLIOGRAFÍA

[1] SIFONTES, C. “Diseño de un sistema de información automatizado para el alquiler de maquinarias de la empresa FERRETECA”. Trabajo de Grado. Universidad de Oriente, Estado Anzoátegui, Venezuela (2004).

[2] MALPICA, M. “Automatización del proceso de acceso a la entrada de maquinarias y vehículos de la empresa Costa Norte Construcciones” Trabajo de Grado. Universidad de Oriente, Estado Anzoátegui, Venezuela (1998).

[3] HERRERA, A. “Diseño de un sistema de control de chequeo y estatus de flota de vehículos adscritos al SETRA”**.** Trabajo de Grado. Universidad de Oriente, Estado Anzoátegui, Venezuela (1998).

[4] QUINTERO, M. "Implantación de un Nuevo Sistema de Control de Inventarios e Investigación Acerca de los Resultados de su Aplicación en la Impsat S.A.” Trabajo de Grado. Universidad Metropolitana (2001)

[**5**] **VELASQUEZ, O.** “Desarrollo de un sistema automatizado para el control de actividades asociadas al área de resguardo para una instalación aduanera en Guanta, Estado Anzoátegui”. Trabajo de Grado. Universidad de Oriente, Estado Anzoátegui, Venezuela (2003).

[**6**] **JACOBSON, I. y BOOCH, G.** “El proceso Unificado de Desarrollo de Software”.Pearson Educación, S.A. Madrid (2000).

[**7**] **ZAVALA, R.** "Ingeniería de Software", Disponible en: www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html (2000).

[**8**] **REY, A. LEAL, M y GONZÁLEZ C**. "Qué es UML?". Disponible en: www.creangel.com/uml/home.html(2.000)

[**9**] **IBM** "Ingeniería de Software" (Manual Técnico) volumen 1. IBM Learning Servicies. Venezuela (2005).

[**10**] **MONTILVA, J.** "Fundamentos y Modelado de Base de Datos Relaciónales", Programa de Estudios de Postgrado en Computación, Universidad de Los Andes, Mérida, Venezuela **(1996)**.

[**11**] **KENDALL, J. Y KENDALL, K. "**Análisis y Diseño de Sistemas**"**. Tercera Edición. Editorial Pearson Educación, México, **(1999)**.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	DESARROLLO DE UN SOFWARE QUE PERMITA LA AUTOMATIZACIÓN PARA LA GESTIÓN DE UNA EMPRESA ARRENDADORA DE VEHÍCULOS
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Arismendi Yaguaraima, Carmelo José	**CVLAC:** 15.035.519 **E MAIL:** carmeloarismendi@gmail.com
De Sia Coppola, Genaro José	**CVLAC:** 16.182.348 **E MAIL:** genarodesia@gmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

Arrendadora

Reservas

Desarrollo

Software

Automatización

Visual Basic Net

Sql Server 2005

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
Ingeniería	Ingeniería en Computación
	Desarrollo de Software
	Automatización

RESUMEN (ABSTRACT):

El presente proyecto consiste en el diseño de una aplicación para automatizar las actividades del proceso de arrendamiento de la empresa Oriente Rent-A-Car C.A, ubicada en el aeropuerto internacional José Antonio Anzoátegui de Barcelona, utilizando la metodología del Proceso Unificado Racional siendo el propósito del mismo, diseñar un sistema automatizado para llevar el control general de los vehículos de la empresa. El proceso de diseño del proyecto, incluyó el levantamiento de información mediante entrevistas realizadas al personal de la empresa, el análisis de la base de datos involucradas, lo cual permitió determinar el comportamiento del sistema actual y así elaborar el modelo del sistema propuesto.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Ing. Aquiles Torrealba	ROL	CA	AS(X)	TU	JU
	CVLAC:	V-7.385.840			
	E_MAIL	artm1966@hotmail.com			
	E_MAIL				
Ing. Rhonald Rodríguez	ROL	CA	AS	TU	JU(X)
	CVLAC:				
	E_MAIL	rhoen2003@hotmail.com			
	E_MAIL				
Ing. Gabriela Veracierta	ROL	CA	AS	TU	JU(X)
	CVLAC:	V-14.616.683			
	E_MAIL	gabrielaveraciertat@hotmail.com			
	E_MAIL				
	ROL	CA	AS	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2010	08	13
AÑO	MES	DÍA

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
tesisArrendadora	.doc

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ (**OPCIONAL**)

TEMPORAL: _____ (OPCIONAL)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:

Pregrado

ÁREA DE ESTUDIO:

Departamento de Computación y Sistemas

INSTITUCIÓN:

Universidad de Oriente. Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

Artículo N° 41. Del Reglamento de Trabajo de Grado.

"Los Trabajos de Grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados a otros fines con el consentimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario"

AUTOR	AUTOR	AUTOR
Carmelo Arismendi	Genaro De Sia	
TUTOR	**JURADO**	**JURADO**
Aquiles Torrealba	Rhonald Rodíguez	Gabriela Veracierta

POR LA SUBCOMISION DE TESIS

José Luis Bastardo