

UNIVERSIDAD ADVENTISTA DE BOLIVIA  
FACULTAD DE INGENIERÍA

SISTEMA WEB PARA EL PROCESO DE PLANIFICACIÓN DE ENVÍO DE CARGA  
Y MONITOREO DE VEHÍCULOS DE LOS SOCIOS DE LA ASOCIACIÓN DE  
TRANSPORTE MIXTO 24 DE SEPTIEMBRE

PROYECTO DE GRADO

Presentado como requisito para obtener  
el grado académico de Licenciado en  
Ingeniería de Sistemas

por

Ronald Martinez Mamani

Tutor

Ing. Víctor Pérez Rojas

Cochabamba, septiembre del 2014



# UNIVERSIDAD ADVENTISTA DE BOLIVIA

## SISTEMA WEB PARA EL PROCESO DE PLANIFICACIÓN DE ENVÍO DE CARGA Y MONITOREO DE VEHÍCULOS DE LOS SOCIOS DE LA ASOCIACIÓN DE TRANSPORTE MIXTO 24 DE SEPTIEMBRE


PROYECTO DE GRADO  
Presentado como requisito para  
Obtener el grado académico de Licenciado en  
Ingeniería de Sistemas

Por

Ronald Martinez Mamani

Formulario de Habilitación para defensa de Modalidad de Graduación aprobada por el  
Ministerio de Educación N° 1617/2014 de fecha 14-11-2014

APROBADA POR EL TRIBUNAL EXAMINADOR

Presidente:   
Mg. Ing. Ivo Alabe Perales  
DECANO FAC. INGENIERIA  
UNIVERSIDAD ADVENTISTA DE BOLIVIA

Examinador interno:   
SIB. MAX ABEL YUCAS MIEZ  
Ingeniero de Sistemas  
RNI 16.764

Examinador Externo:   
SIB. EFRAIN FERNANDO  
LUNA MAMANI  
Ingeniero de Sistemas  
RNI 29.222

Examinador interno:   
SIB. ROMULO RICKMAN HUISE  
Ingeniero de Sistemas  
RNI 22.614

5-03-2015  
Fecha de Aprobación:

16:50 p.m.  
Hora de aprobación:

Los firmantes manifiestan la aprobación de este trabajo de investigación satisface los requerimientos establecidos.

Las ideas expresadas en la investigación son responsabilidad exclusiva de autor y no reflejan el pensamiento sustentado por la Universidad Adventista de Bolivia.

Realizamos tus sueños

www.uah.edu.bo

## **DEDICATORIA**

El presente proyecto está dedicado a mis padres Ricardo Martinez y Bertha Mamani, a la compañera que Dios me dio para esta vida Raquel M. Aguilar y a mis hijos Josias Emmanuel y Keyler Gael que supieron comprender la ausencia en el hogar por alcanzar este objetivo.

## **AGRADECIMIENTOS**

A Dios, por darme la sabiduría y fuerza para realizar este proyecto.

A mis queridos hermanos: José, Aydé, Cimar y Dennis que me apoyaron en cada instante de mi carrera.

A mis padres, por brindarme el apoyo durante la carrera y hacer realidad mis sueños.

A mis docentes de la Universidad que con paciencia supieron proporcionarme una excelente formación.

## **RESUMEN**

Los sistemas de información brindan una gran ayuda en las instituciones, porque facilitan la administración de manera diferente y eficaz los procesos que realizan en cada gestión a lo largo del ciclo de vida. Esto con el objetivo de que exista un mejor desenvolvimiento y rendimiento de las personas que están involucradas dentro organizaciones.

El presente proyecto nace de la necesidad de sistematizar el proceso asignación y envío de carga e implementando tecnología GPS para el monitoreo de los vehículos, en la Asociación de Transporte Mixto 24 de Septiembre.

Para el desarrollo de software se ha planteado utilizar la plataforma de desarrollo de software Microsoft Visual Studio, y como gestor de base de datos SQL Server 2008, para el uso seguro de los usuarios del sistema, seleccionar el modelos Programación Extrema (XP) para la dirección del proyecto, que permitirá brindar el desarrollo de trabajo de manera estrecha con los usuarios.

A la vez se utilizó tecnologías GPS para la implementación del módulo del control vehicular, desarrollando un subsistema llamado Soket como programa que nos permitirá escuchar las trazas enviadas por el receptor al servidor.

Por último se implementó API de Google Maps para la gráfica de las trazas enviadas por el receptor.

**Palabras clave:** Monitoreo Vehicular - sistema GPS – tecnologías GPS.

## ÍNDICE

Pág.

INTRODUCCIÓN .....	1
CAPÍTULO 1.....	3
EL PROBLEMA.....	3
1.1 Antecedentes.....	3
1.1.1 Antecedentes organizacionales .....	3
1.1.2 Antecedentes tecnológicos.....	5
1.2. Problema .....	5
1.2.1 Situación Problemática.. .....	6
1.2.2 Formulación del problema.. .....	7
1.3 Objetivos.....	7
1.3.1 Objetivo General.....	7
1.3.2 Objetivos específicos.. .....	8
1.4 Justificación .....	8
1.5 Límites y Alcances.....	8
1.5.1 Límites .....	8
1.5.2 Alcances.....	9
1.6 Estudio de factibilidad .....	9
1.6.1 Factibilidad técnica .....	9
1.6.2 Factibilidad económica .....	10
1.6.3 Factibilidad operacional.....	12
CAPÍTULO 2.....	13
MARCO TEÓRICO. ....	13
2.1 Modelado de negocios... .....	13
2.1.1 Modelos de análisis.....	13
2.1.1.1 Dominio de la información... .....	14
2.1.1.2 Dominio funcional... .....	14
2.1.1.3 Dominio de comportamiento... .....	14
2.1.2 Modelos de diseños.....	14
2.1.3 Definición de proceso... .....	15
2.1.3.1 Elementos de un proceso... .....	15
2.1.4 Diagrama de actividades... .....	15
2.2 Sistema GPS... .....	17
2.2.1 Tecnología GPS... .....	17
2.2.2 Composición del sistema GPS.....	18
2.2.2.1 Segmento espacial.....	18
2.2.2.2 Segmento de control. ....	18
2.2.2.3 Segmento de usuario.....	19
2.2.3 Tipos de GPS. ....	20
2.2.3.1 Navegadores convencionales.....	21
2.2.3.2 Receptores de C/A avanzados.....	22
2.2.3.3 El método diferencial DGPS.....	23
2.2.3.4 Elección de método de receptor GPS .....	24

2.2.4 Receptores GPS...	24
2.2.4.1 Funcionalidad GPS...	25
2.2.5 Sistema GPRS.....	25
2.2.5.1 GSM.....	26
2.2.5.2 GPRS.....	26
2.2.6 Dispositivos GPS.....	27
2.2.6.1 Elección del receptor GPS.....	29
2.2.6.2 Funciones del receptor MVT340.....	29
2.2.6.3 Accesorios de un eceptor.....	30
2.2.6.4 Características de un receptor MVT340.....	31
2.3 Proceso de envío de datos de un receptor.....	33
2.3.1 Definición y características de un SOKET.....	34
2.3.2 Propiedades de un SOKET.....	35
2.3.3 Atributos de un SOKET.....	35
2.3.4 Tipos de SOKET.....	35
2.4 Ingeniería de software.....	36
2.4.1 Procesos de desarrollos de un software clásicos.....	36
2.4.2 Métodos de desarrollo ágil.....	37
2.4.2.1 Scrum.....	37
2.4.2.2 Programación extrema XP.....	38
2.4.2.3 Proceso unificado ágil (PUA).....	38
2.4.2.4 Elección de la metodología de desarrollo.....	40
2.4.3 Programación extrema.....	41
2.4.3.1 Facetas de la vida de un proyecto con XP.....	42
2.4.3.2 Exploración.....	42
2.4.3.3 Planificación de la entrega.....	42
2.4.3.4 Iteraciones.....	43
2.4.3.5 Producción.....	43
2.4.3.6 Mantenimiento.....	44
2.4.3.7 Muerte del proyecto.....	44
2.4.4 Lenguaje Unificado de modelado (UML).....	44
2.4.4.1 Diseño UML en XP.....	44
2.4.4.2 Diagramas en clases.....	45
2.4.4.3 Diagramas de iteración.....	46
2.4.5 Enterprise architect como herramienta para UML.....	47
2.5 Patrón de arquitectura de software.....	50
2.5.1 Patrón de diseño modelo vista controlador.....	50
2.5.2 MVC con la arquitectura N-capas.....	52
2.5.3 Arquitectura N-capas.....	52
2.5.3.1 La capa de presentación.....	53
2.5.3.2 Capa de aplicación.....	54
2.5.3.3 Capa de lógica.....	54
2.5.3.4 Capa de acceso a datos.....	54
2.6 Herramienta de programación.....	55
2.6.1 Plataforma .NET.....	55
2.6.1.1 Microsoft visual studio 2012.....	56
2.6.1.2 ASP .NET MVC4.....	57

2.6.1.3 Razor: motor de vista para APS .net.....	58
2.7 Gestores de base de datos.. .....	59
2.7.1 Microsoft SQL server 2008 express.. .....	59
2.7.2 Elección de gestión de bases de datos .....	60
2.7.3 Entity framerork: code firts.....	60
2.8 Seguridad.. .....	61
2.8.1 Confidencialidad.....	62
2.8.2 Integridad.....	62
2.8.3 Disponibilidad.....	62
2.8.4 Autenticidad.....	62
2.9 Calidad de software .....	63
2.10 Pruebas unitarias .....	63
2.10.1 Unit Test .....	64
 CAPÍTULO 3.....	66
MARCO PRÁCTICO. ....	66
3.1 Modelado de negocios .....	66
3.1.1 Modelado de negocio actual .....	66
3.1.2 Modelado de negocio alternativo.....	71
3.2 Planificación de iteración para el desarrollo de software .....	78
3.3 desarrollar un módulo para la gestión de vehículos, socios y choferes .....	79
3.3.1 Historias de usuario .....	79
3.3.1.1 Historia de usuarios registros de socios .....	79
3.3.1.2 Historia de usuarios gestión de choferes.....	80
3.3.1.3 Historia de usuarios registros de vehículos.....	80
3.3.1.4 Historia de usuarios gestión de tipo de vehículo .....	81
3.3.1.5 Historia de usuarios gestión observaciones .....	82
3.3.1.6 Historia de usuarios gestión de cuentas de usuarios .....	82
3.3.1.7 Historia de usuarios gestión de requisitos.....	83
3.3.2 Planificación .....	84
3.3.2.1 Estimación de esfuerzo .....	84
3.3.3 Diagrama de caso de uso general primera iteración .....	86
3.3.3.1 Diagramas de colaboración.....	87
3.3.3.2 Diagrama de colaboración gestionar chofer .....	87
3.3.3.3 Diagrama de colaboración gestión vehículos .....	88
3.3.3.4 Diagrama de colaboración gestión de tipo de vehículo .....	88
3.3.3.5 Diagrama de colaboración gestión de observadores.....	89
3.3.4 Diseño .....	89
3.3.4.1 Diseño físico de la base de datos de la primera iteración .....	90
3.3.5 Arquitectura del sistema .....	91
3.3.6 Codificación.....	92
3.3.6.1 Modelo .....	93
3.3.6.2 Controlador .....	93
3.3.6.3 Vista .....	95
3.3.7 Pruebas.....	98
3.3.7.1 Pruebas unitarias .....	98
3.4 Segunda iteración gestión de asignación de envío de carga .....	100



3.4.1 Historia de usuario .....	100
3.4.1.1 Historia de usuario gestión carga.....	100
3.4.1.2 Historia de usuario asignación de carga .....	100
3.4.2 Planificación de entrega.....	101
3.4.3 Diagrama de caso de uso general segunda iteración.....	102
3.4.3.1 Diagrama de colaboración gestionar carga .....	103
3.4.3.2 Diagrama de colaboración asignación de carga.....	103
3.4.4 Diseño .....	104
3.4.4.1 Diseño físico de la base de la segunda iteración .....	104
3.4.5 Codificación .....	106
3.4.5.1 Modelo .....	106
3.4.5.2 Controlador .....	106
3.4.5.3 Vista .....	109
3.4.6 Pruebas.....	110
3.4.6.1 Pruebas unitarias .....	110
3.5 Tercera iteración monitoreo vehicular .....	111
3.5.1 Historia de usuario .....	111
3.5.1.1 Historia de usuario Gestión de receptores .....	111
3.5.1.2 Historia de usuario Control de monitoreo.....	112
3.5.2 Planificación de entrega.....	113
3.5.3 Diagrama de caso de uso general tercera iteración.....	114
3.5.3.1 Diagramas de colaboración gestionar receptor .....	115
3.5.3.2 Diagramas de colaboración asignación de carga .....	115
3.5.4 Diseño .....	116
3.5.4.1 Diseño físico de la base de datos de la tercera iteración.....	117
3.5.5 Codificación.....	118
3.5.5.1 Modelo .....	118
3.5.5.2 Controlador .....	119
3.5.5.3 Vista .....	119
3.5.6 Implementación del subsistema Somet.....	121
3.5.7 Configuración del receptor GPS .....	121
3.5.7.1 Implementación API de Google Maps en el sistema.....	123
3.5.5.1 Esquema General de la implementación de la tecnología GPS en el sistema .....	124
CONCLUSIONES .....	126
RECOMENDACIONES.....	127

## ÍNDICE DE FIGURAS

Figura 2.1 Diagrama de actividades. ....	16
Figura 2.2 Constelación NAVSTAR. ....	18
Figura 2.3 Estación de control NAVSTAR. ....	19
Figura 2.4 Segmento de usuario. ....	20
Figura 2.5 Navegadores convencionales. ....	22
Figura 2.6 Receptores GPS C/A Avanzados. ....	23
Figura 2.7 Accesorios de un receptor. ....	31
Figura 2.8 Envío de datos de un receptor al servidor. ....	33
Figura 2.9 Captura de una traza del receptor GPS. ....	34
Figura 2.10 Preferencia de modelos ágiles. ....	41
Figura 2.11 Facetas de un proyecto en programación Extrema. ....	42
Figura 2.12 Diagramas de clases. ....	46
Figura 2.13 Diagrama de colaboración. ....	47
Figura 2.14 Diagrama de la arquitectura patrón MVC. ....	51
Figura 2.15 Arquitectura N-Capas. ....	53
Figura 2.16 Relación entre varias tecnologías web ASP .NET. ....	57
Figura 2.17 Plataformas de datos SQL Server. ....	59
Figura 3.1 Modelado de Negocios Actual. ....	67
Figura 3.2 Modelado de Negocio Alternativo. ....	71
Figura 3.3 Gestionar socio. ....	73
Figura 3.4 Gestionar chofer. ....	74
Figura 3.5 Gestionar vehículo. ....	75
Figura 3.6 Habilitación de vehículo. ....	76
Figura 3.7 Asignación de carga. ....	77
Figura 3.8 Diagrama de caso de uso primera iteración. ....	86
Figura 3.9 Diagrama de colaboración gestión de socio. ....	87
Figura 3.10 Diagrama de colaboración gestión de chófer. ....	87
Figura 3.11 Diagrama de colaboración gestión vehículos. ....	88
Figura 3.13 Diagrama de colaboración gestión de observaciones. ....	89
Figura 3.14 Modelo entidad de relación. ....	90
Figura 3.15 Diseño físico del sistema para la primera iteración. ....	91
Figura 3.16 Implementación de la arquitectura en la primera iteración. ....	91
Figura 3.17 Modelo administrar vehículo. ....	93
Figura 3.18 Método crear vehículo. ....	94
Figura 3.19 Método editar vehículo. ....	94
Figura 3.20 Método eliminar vehículo. ....	95
Figura 3.21 Vista de creación de vehículo. ....	96
Figura 3.22 Vista de lista de vehículos. ....	96
Figura 3.23 Vista de creación de socio. ....	97
Figura 3.24 Vista de lista de socios. ....	98
Figura 3.25 Pruebas de la primera iteración. ....	99
Figura 3.26 Resultado de prueba de primera iteración. ....	99
Figura 3.27 Diagrama de colaboración segunda iteración. ....	102
Figura 3.28 Diagrama de colaboración gestión de carga. ....	103

Figura 3.29 Diagrama de colaboración asignación de carga .....	103
Figura 3.30 Modelo de identidad de relación segunda iteración .....	104
Figura 3.31 Diseño físico del sistema para la segunda iteración .....	105
Figura 3.32 Modelo de asignación de carga .....	106
Figura 3.33 Controlador método asignar carga .....	107
Figura 3.34 Vista de lista de cargas .....	109
Figura 3.35 Vista asignación de carga .....	109
Figura 3.36 Test método asignar carga .....	110
Figura 3.37 Resultado de prueba unitaria del método asignar carga .....	111
Figura 3.38 Diagramas de caso de uso segunda iteración .....	114
Figura 3.39 Diagrama de colaboración gestionar traza .....	115
Figura 3.40 Diagrama de colaboración Gestionar recepto .....	113
Figura 3.41 Modelo entidad de relación tercera iteración .....	116
Figura 3.42 Diseño físico del sistema para la segunda iteración .....	117
Figura 3.43 Modelo de asignación de carga .....	118
Figura 3.44 Controlador método crear receptor .....	119
Figura 3.45 Vista de crear receptor .....	120
Figura 3.46 Vista de lista de receptor .....	120
Figura 3.47 Vista panel de configuración de receptor GPS .....	121
Figura 3.48 Configuración de la Ip y puerto en el Scket .....	122
Figura 3.49 Código de almacenamiento de la traza en la base de datos .....	123
Figura 3.50 Implementación de Google Maps .....	124
Figura 3.51 Esquema de diseño de implementación y envió de datos del dispositivo GPS al servidor .....	125

## ÍNDICE DE TABLAS

Tabla 1.1 Comparación de precio al adquirir el servicio de Bolivia-GPS primer año. ....	11
Tabla 1.2 Gasto al implementar servicio GPS. ....	11
Tabla 2.1 Símbolos en los diagramas de actividades. ....	16
Tabla 2.2 Tabla de comparación de receptores. ....	27
Tabla 2.3 Características de un receptor mvt340. ....	31
Tabla 2.4 Tipos de Sockets. ....	35
Tabla 2.5 Cuadro de comparación de las metodologías ágiles y tradicionales. ....	39
Tabla 2.6 Tabla de comparación de herramientas UML. ....	48
Tabla 3.1 Especificaciones de módulos e iteraciones. ....	78
Tabla 3.2 Historia de usuario Gestión de socios. ....	79
Tabla 3.3 Historia de usuario gestión de choferes. ....	80
Tabla 3.4 Historia de usuario gestión de vehículos. ....	80
Tabla 3.5 Historias de usuario gestión de tipo vehículos ....	81
Tabla 3.6 Historia de usuario gestión observaciones. ....	82
Tabla 3.7 Historia de usuario gestión de cuentas de usuario. ....	82
Tabla 3.8 Historia de usuario gestión de cuentas de usuarios. ....	83
Tabla 3.9 Modelo para estimación de tiempo. ....	85
Tabla 3.10 Estimación de tiempo a la primera iteración. ....	85
Tabla 3.11 Historia de usuario registro de carga. ....	100
Tabla 3.12 Historia de usuario registro de socio. ....	100
Tabla 3.13 Estimación de esfuerzo de la segunda iteración. ....	101
Tabla 3.14 Historia de usuario gestión de receptores. ....	111
Tabla 3.15 Historia de usuario control de monitoreo. ....	112
Tabla 3.16 Estimación de esfuerzo de la tercera iteración. ....	113

## INTRODUCCIÓN

En los últimos tiempos el avance de la tecnología ha involucrado el desarrollo de nuevas herramientas que utiliza el hombre para ahorrar tiempo, esfuerzos y en algunos casos dinero gracias a las innovaciones que se han repercutido en la sociedad.

En la actualidad, es difícil imaginar un mundo sin computadoras. La idea de una sociedad totalmente informatizada, muchos consideraban esto una fantasía que con el pasar del tiempo se transformó en una realidad, a medida que la tecnología avanza fue tomando lugar en distintos ámbitos de la ciencia y la vida cotidiana, a la vez brindando beneficios indirectamente debido a que realizan usos de los servicios de las compañías y empresas que usan diariamente la computadora.

El contar con fuentes de información confiable veraz y oportuna que ofrezcan un criterio para la toma de decisiones acertadas para beneficio de la empresa es lo que quieren las empresas y sobre todo contar con un sistema de información que sean fáciles en el manejo, en este trabajo se expone el caso de la Asociación de transportes mixto 24 de septiembre, ubicado en el municipio de Capinota Cochabamba, la cual se dedica al servicio de transporte de envío de carga a la planta de producción de Coboce.

En el primer capítulo se observa con mayor detalle el proceso de envío de carga y los problemas que se notaron en el proceso de asignación de cargas y monitoreo de los vehículos de los respectivos socios. Para ello se da una solución lo cual sería implementando un sistema como ayuda en los procesos mencionados.

Para continuar se tiene un segundo capítulo donde se da un respaldo teórico sobre las tecnologías a implementarse, metodologías herramientas a utilizarse obteniendo un panorama claro de lo que se pretende utilizar.

Pasando al tercer capítulo se puede notar el análisis formulado, a partir de datos brindados por parte del presidente de la asociación para tener un panorama claro del sistema a desarrollarse.

## **CAPÍTULO 1**

### **EL PROBLEMA**

En este apartado se explicara el proceso con el cual trabaja la organización y su problemática mostrando un panorama general del perfil del proyecto de grado, se tratará los siguientes temas: antecedentes, problemas, objetivos, alcances, límites, justificación y viabilidad.

#### **1.1 Antecedentes**

A continuación se describirán los antecedentes organizacionales y tecnológicos que tiene la institución para el proceso de asignación y envío de carga en la Asociación de transporte mixto 24 de septiembre.

##### **1.1.1 Antecedentes organizacionales**

En la actualidad Bolivia cuenta con una variedad de empresas dedicadas a la producción de cementos tales como: COBOCE, SOBOCE, FRANCESA, cemento WARNES, etc.

Para el envío de sus productos a distintos lugares del país utilizan el servicio de asociaciones de transportes dedicadas al transporte pesado.

COBOCE trabaja con tres asociaciones de transportes de las cuales nos interesa saber más a detalles una de ella ASOTRAM “Asociación de Transporte Mixto 24 de Septiembre”.

ASOTRAM fue fundada el 1 de mayo del 2005 con una cantidad de 100 socios desde la fecha hubo un incremento casi del 200% esto significa que ahora cuenta con 295 socios, están clasificados como activos, pasivos e inactivos.

Los socios activos son los que prestan servicio de transporte permanente, los pasivos son los que conforman el directorio y los inactivos son socios que cuentan con una licencia temporánea.

La oficina de ASOTRAM se encuentra en Irpa Irpa, Capinota. La institución no cuenta con un inmueble propio por lo cual brinda sus servicios desde una oficina en alquiler. Esta oficina está dividida en tres secciones: sala de recepción, archivos y administrativa.

Los archivos de los socios, están almacenados alfabéticamente en un estante que la institución tiene.

A continuación se describirán en detalle los procedimientos que realiza ASOTRAM para habilitación y monitoreo de vehículos.

COBOCE solicita una determinada cantidad de vehículos para realizar el despacho de carga indicando la fecha, hora, cantidad de carga a llevar, y el destino. La secretaria de ASOTRAM anota la petición y revisa el listado de vehículos disponibles.

Cuando un socio solicita la habilitación de su vehículo para ser incluido en el rol de reparto de carga la secretaria necesita verificar algunos aspectos del socio tales como sanciones, infracciones, cuotas, etc.

El llenado del formulario de salida se realiza manualmente.

Cuando un socio incumple una norma de la institución, la secretaria realiza el llenado manualmente en la ficha de infracciones y luego procede al guardado en el Folder del socio.

Una de las infracciones más comunes es cuando el vehículo llega tarde a la planta de COBOCE para el despacho de carga.

Cuando se requiere un reporte de los socios que están en servicio, para saber cuántos vehículos disponibles tendrá en la semana, la secretaria tiene que realizar manualmente el revisando del folder de salida semanales.



Los procedimientos actuales en el monitoreo de vehículos se realiza por la vía telefónica y los datos brindados por los choferes en algunos casos no son ciertas.

En ocasiones la institución recibe llamadas telefónicas de COBOCE indicando que la carga enviada aún no ha llegado al destino porque la demora es de mucho tiempo.

Existen algunos casos en que los socios o la institución necesitan saber dónde exactamente esta su vehículo y para ello proceden a los recursos de la vía telefónica para contactarse con el chofer, pero en algunos casos la comunicación no se hace efectiva.

### **1.1.2 Antecedentes tecnológicos**

GPS (Sistemas de Posicionamiento Global) es el primer sistema de navegación basada en satélites, que entra en servicio en al 1965, implementado por los departamentos de defensa, transporte de la agencia espacial norteamericana (DoD, DoT y NASA).

El objetivo de la implementación de este sistema era de contar con un sistema que sea capaz de determinar la localización basada en satélites.

Desde la fecha hasta hoy la implementación de estos sistemas ha tomado un lugar muy importante en las empresas que brindan servicios de seguridad y transporte ya sean terrestres, acuáticos y aéreas.

Por otro lado en Bolivia también juega un papel muy importante desde la aprobación del decreto **Nº 1366**, el 3 de octubre del 2012 como parte del plan “vivir con seguridad” que se desglosa de la siguiente manera “Que el Parágrafo V del Artículo 238 de la Ley Nº 165, de 16 de agosto de 2011, Ley General de Transporte, establece que de manera prioritaria se deberá implementar en el servicio público de transporte automotor terrestre, el monitoreo mediante Sistemas de Posicionamiento Global - GPS u otro dispositivo electrónico de similares características y hojas de ruta electrónicas para el control efectivo del equipo de conducción y vehículos en terminales terrestres y trancas intermedias”.

El mencionado decreto fue emitido porque en los últimos años se vio una cantidad elevada de muertes y accidentes en el transcurso de viajes interdepartamentales, es por eso que la “Autoridad de Regulación y Fiscalización de telecomunicación y transporte” (ATT) es encargada de colocar estos dispositivos en los buses de transportes interdepartamentales.

El sistema GPS es implementado con el fin de tener los datos precisos sobre el estado de la flota si esta estática o en movimiento, verificar la velocidad, datos de los pasajeros, hora de salida.

Por otro lado en Bolivia hay instituciones que brindan este servicio de instalación de sistemas GPS tales como Bolivia-GPS es una institución que brinda este tipo de servicio sobre territorio nacional, para adquirir este servicio los precios que se estiman son de 250\$us para los automóviles particulares sin fines de lucro y hasta un precio tope de 200\$us a instituciones con fines de lucro. Este precio es el pago del primer mes dando soporte a la instalación del dispositivo de receptor GPS.

Luego del primer mes los usuarios que cuentan con la respectiva instalación deben realizar un pago mensual de 20 \$us por el uso de la plataforma y la cuenta establecida.

## **1.2 Problema**

En este apartado se definirá la situación problemática que tiene la organización, para luego formular un problema principal el cual será resuelto con el siguiente proyecto.

### **1.2.1 Situación problemática**

A continuación se describen los problemas encontrados:

- La verificación de infracciones y faltas cometidas se realiza manualmente ocasionando demora en el proceso para la habilitación de vehículos.
- La asignación del envío de carga se realiza según el orden de habilitación tomándose en cuenta simplemente la disponibilidad provocando así desconformidad en algunos socios.

- El llenado del formulario de despacho de carga se realiza manualmente provocando ilegibilidad y errores en los datos.
- El control para el acceso a la sala de archivos es ineficiente, esto ocasiona pérdida de documentos.
- El proceso de comunicación con los choferes para saber la ubicación exacta del vehículo es ineficiente ya que se sabe que algunos lugares no cuentan con coberturas telefónicas para la comunicación, esto ocasiona preocupación a la institución y a los socios.
- Los datos brindados por los choferes por la vía telefónica sobre la ubicación pueden no ser ciertas ya que algunos acostumbran realizar transporte de cargas en el proceso de retorno sin el conocimiento del propietario del vehículo, esto ocasiona pérdida económica de los socios.
- La adquisición de servicios GPS mediante empresas que proveen este servicio, tiene un costo elevado y esto provocaría una inversión económica elevada para los socios de la institución.

### **1.2.2 Formulación del Problema**

Los procedimientos manuales aplicados en el proceso de planificación del envío de carga y monitoreo de vehículos de los socios de la Asociación de Transporte Mixto 24 de Septiembre provocan un menor número de entregas de carga y pérdidas económicas.

## **1.3 Objetivos**

Para la solución de los problemas mencionados anteriormente, a continuación se plantea el objetivo general y los objetivos específicos.

### **1.3.1 Objetivo general**

Desarrollar un sistema web para el proceso de planificación de envío de carga y monitoreo de vehículos de los socios de la Asociación de transporte mixto 24 de septiembre utilizando la tecnología GPS.

### **1.3.2 Objetivos específicos**

- Diseñar un modelo de negocio alternativo para el proceso de planificación y monitoreo de vehículos.
- Implementar el subsistema de gestión de vehículos, dueños y choferes.
- Desarrollar el módulo para la integración de tecnología GPS para monitoreo de los vehículos.
- Implementar mecanismos para la habilitación y disponibilidad de vehículos.
- Diseñar un algoritmo para la optimización en el proceso de distribución de carga.

### **1.4 Justificación**

La justificación para el sistema de planificación de envío de carga y monitoreo de vehículos se realiza de acuerdo a los beneficios que le brindara el sistema

- Los datos almacenados de manera física se las podrán tener de manera digital esto ayudara mucho al momento de realizar reportes.
- Con la implementación del sistema se podrá realizar de una manera más eficiente el proceso de habilitación para el despacho de carga.
- Los usuarios podrán ver los reportes necesarios desde cualquier lugar sin tener que aproximarse a las oficinas de la institución.
- Los socios y la institución podrán realizar la consulta sobre la ubicación exacta del vehículo solicitado, a través de su cuenta personal en el sistema.

### **1.5 Límites y alcances**

EL presente proyecto presentara límites en cuanto a desarrollo y alcances en funcionalidades las cuales se describen a continuación.

#### **1.5.1 Límites**

A continuación se describen los límites del sistema

- El sistema tendrá la capacidad de realizar el proceso de habilitación y monitoreo de los vehículos. No se implementara en el sistema el área de contable que la institución maneja.
- El sistema podrá monitorear los vehículos siempre y cuando se encuentre dentro del territorio Boliviano.

### **1.5.2 Alcances**

A continuación se describe el alcance del sistema.

- Dentro de los alcances del sistema de habilitación, se podrá tener un registro minucioso sobre las infracciones y faltas cometidas por los socios y vehículos.
- Con el sistema propuesto la administración podrá tener un informe detallado sobre los servicios prestado por cada socio.
- La administración y socios podrán saber la ubicación exacta de los vehículos cuando sea necesario consultar.

## **1.6 Estudio de factibilidad**

### **1.6.1 Factibilidad técnica**

Para el desarrollo de la del presente proyecto se utilizara las tecnologías y herramientas siguientes:

- Plataforma de C# .NET Framework 4, Visual Studio 2012.
- Gestor de base de datos SQL Server 2008.
- Implementando la arquitectura MVC.

Características del ordenador que será usado para el desarrollo del proyecto:

- Sistema Operativo Windows 8 Enterprise de 64 bits.
- Memoria Ram 8Gb.
- Procesador Intel core i7.

Para la recepción de datos se usara el GPS MVT340 los servicios que brinda este dispositivo son:

- **Red:** GSM / GPRS

- **Banda:** 900/1800/1900 MHz o 850/900/1800 / 1900MHz
- **Chip GPS:** SIRF 3 CHIP
- **Sensibilidad del GPS:**-159dBm
- **Precisión del GPS:** 5m
- **Tiempo al primer fijo:**
  - Readquisición: 0,1 s
  - Estado frío: 45s
  - Estado tibio: 35s
  - Estado caliente: 1s
- **Voltaje de trabajo:** DC 9 ~ 30V
- **Batería cambiable 3.7V / 1.2A Li-ion:** Batería

Para el desarrollo del proyecto propuesto se puede observar que se tiene con los conocimientos necesarios para iniciar con desarrollo, utilizando dichas tecnologías.

### **1.6.2 Factibilidad económica**

Para la adquisición de las licencias de las herramientas a usar, ASOTRAM optará por el alquiler de servidor para la plataforma .NET, una de las instituciones que brinda este servicio es FEROZO es una empresa Mexicana el costo por el servicio es de 800 Bs. Anual, con la adquisición de este servicio la institución no tendrá que preocuparse de obtener las licencias para el desarrollo en las herramientas de Microsoft.

Para la obtención de los receptores GPS la institución no hará ninguna inversión ya que se entró en un acuerdo de que cada socio realizará esta inversión que tendrá un costo de 80\$us para la obtención del receptor.

A continuación se realizara una comparación de la inversión que tiene que realizar la institución al adquirir el servicio de Bolivia-GPS.

**Tabla 1.1**  
**Comparación de precio al adquirir el servicio de Bolivia-GPS primer año**

<b>Cantidad de socios</b>	<b>Precio mensual por el uso del plataforma</b>	<b>Precio/Anual Por el uso de la plataforma</b>	<b>Precio receptor</b>	<b>Inversión Total</b>
1	20\$us	240\$us	200\$us	440\$us
290	5800\$us	69600\$us	58000\$us	127600

**Fuente.** Elaboración propia

En la tabla superior se puede observar el precio establecido como inversión por socio y como institución. Como socio tendría que invertir 440\$us el primer año luego del primer año tendría que invertir 240\$us. Porque luego del primer año solo cancelaria el servicio por el uso de la plataforma Cada socio.

A continuación se realiza la cotización de inversión de cada socio al implementar el sistema en la institución.

**Tabla 1.2**  
**Gasto al implementar servicio GPS**

<b>Nº de socio</b>	<b>Paquete gprs /mes Tarifa Sim</b>	<b>Precio/Anual paquete de datos ENTEL</b>	<b>Precio receptor</b>	<b>Inversión Total</b>
1 Socio	5\$us.	60\$us	80\$us	140\$us
290 Socios	1450\$us	17400\$us	23200\$us	40600\$us

**Fuente.** Elaboración propia

Al desarrollar el sistema GPS para la institución cada socio tendría que invertir un monto de 140\$us el primer año, luego del primer año solo tiene que pagar 60\$us en la recarga de crédito al Sim para el envío de datos.

### **1.7 Factibilidad operacional**

Con la identificación de los antecedentes y situación problemática descrita en las respectivas secciones se ha podido identificar los procesos y dificultades que se tiene al momento de realizar algunos procesos.

Por este motivo la implementación del sistema en la “Asociación de transporte mixto 24 de septiembre” permitirá sistematizar los procesos asignación del envío de carga, rastreo vehicular y algunas operaciones de manera eficiente.

En cuanto a los procesos e información para su eficiencia se tiene como objetivo trabajar mutuamente con los socios, secretaria y presidente de la asociación. Además gracias al plan de trabajo tiene gran aceptación entre los que van a utilizar por parte de la directiva por parte de la directiva y el resto de los socios que piensan que el sistema es ideal para sus metas y que el problema será superado.



## **CAPÍTULO 2**

### **MARCO TEÓRICO**

Este capítulo se promocionara información teórica fundamental de los diferentes tipos de herramientas para la recolección de información necesaria, el análisis, diseño y herramientas para realizar la programación.

#### **2.1 Modelado de Negocios**

El modelado de negocios tiene como finalidad describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocios.

Con el modelado de negocio se dará a conocer de mejor forma como opera el sistema de habilitación de vehículos y envío de carga en la “Asociación de transporte mixto 24 de septiembre”.

Modelar el proceso de Negocio es una parte esencial de cualquier proceso de desarrollo de software. Permite al analista capturar el esquema general y los procedimientos que gobiernan el negocio. Este modelo provee una descripción de donde se va a ajustar el sistema de software considerado dentro de la estructura organizacional y de las actividades habituales. También provee la justificación para la construcción del sistema de software al capturar las actividades manuales y los procedimientos automatizados habituales que se incorporaran en nuevo sistema, con costos y beneficios asociados [8].

Pressman [6], menciona que en el trabajo de la ingeniería del software se crean dos clases de modelos: modelos de análisis y modelos de diseño:

##### **2.1.1 Modelos de análisis**

Representan los requisitos del cliente al presentar el software en tres dominios diferentes: el dominio de la información, el dominio funcional y el dominio del comportamiento.

#### 2.1.1.1 Dominio de la información

Lo forman los datos que fluyen hacia el sistema (a partir de los usuarios finales, otros sistemas o dispositivos externos), los datos que fluyen desde el sistema (a través de la interfaz del usuario, interfaces de red, reportes, gráficas y otros medios) y los almacenamientos de datos que se recopilan y reorganizan los objetos consistentes de información (es decir, los datos que se mantienen en forma permanente).

#### 2.1.1.2 Dominio Funcional

Las funciones del software proporcionan un beneficio directo a los usuarios finales y también aporta soporte interno a aquellas características visibles para el usuario.

#### 2.1.1.3 Dominio del Comportamiento

La entrada que proporcionan los usuarios finales, los datos de control que aporta un sistema externo o los datos de monitoreo que se recolectan a través de una red ocasionan que el software se comporte de una manera específica.

### 2.1.2 Modelos de diseños

Los modelos de diseño representan características del software que ayudan a los profesionales a construirlos de manera efectiva: la arquitectura, la interfaz del usuario, y el detalle al nivel de componentes.

**La arquitectura.** Es el esqueleto del sistema que se va construir. Este afecta las interfaces, las estructuras de datos, el flujo y el comportamiento del control del programa.

**La interfaz del usuario.** Es la manifestación visible del software, un interfaz de diseño pobre siempre conduce a la percepción de que el software está mal hecho.

**El detalle a nivel de componentes.** La independencia funcional es una medida del “significado sencillo” de un componente del software. La funcionalidad que entrega un componente debe ser cohesiva; es decir, debe centrarse en una función o sub función.

### 2.1.3 Definición de Proceso

Weitzenfeld lo define de la siguiente manera “un proceso es un conjunto de actividades<sup>1</sup> mutuamente relacionadas o que interactúan, las cuales transforman elementos de entradas en resultados” [9].

#### 2.1.3.1 Elementos de un proceso.

Todo proceso tiene tres elementos: Input, Secuencia de Actividades y Output. A continuación se explicaran cada una de ellas.

**Input**, es un elemento que proviene de un suministrador (externo o interno), es la salida de otro proceso (precedente en la cadena de valor) o de un proceso del proveedor o del cliente.

La **Secuencia de Actividades**, son entradas laterales, es decir, inputs necesarios o convenientes para la ejecución del proceso, pero cuya existencia no lo desencadena.

Un **Output**, la salida es un producto que va destinado a un usuario o cliente (externo o interno); el output final de los procesos de la cadena de valor es el input o una entrada para un proceso del cliente

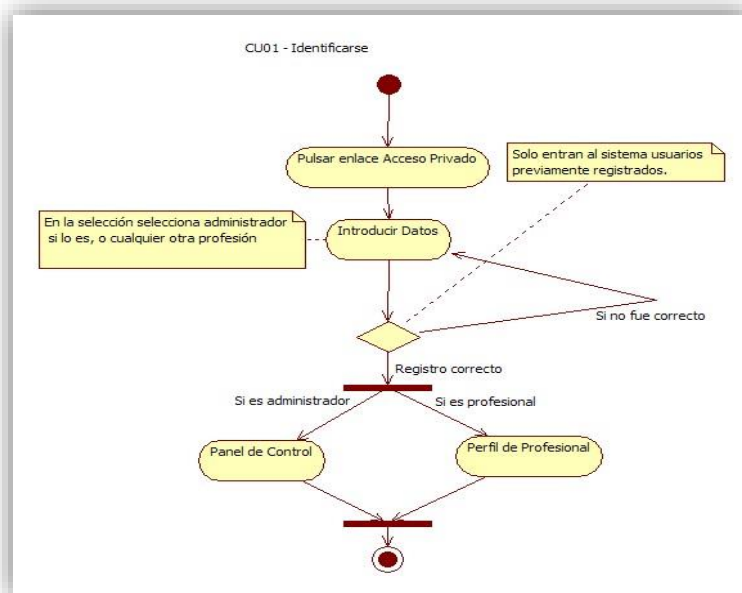
### 2.1.4 Diagrama de Actividades

Los diagramas de actividades permiten describir como un sistemas (empresa, instituciones, etc.) implementa su funcionalidad, modela el comportamiento, transacciones. [10]

---




<sup>1</sup> Actividades: Conjunto de tareas para obtener un resultado.

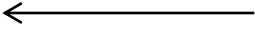
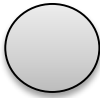
**Figura 2.1**  
**Diagramas de actividades**



**Fuente.** [10, fig. 3.4]

**Tabla 2.1**  
**Símbolos en los diagramas de actividades**

Figura	Descripción
	Se suele utilizar este símbolo para representar el origen de una entrada o el destino de una salida. Se emplea para expresar el comienzo o el fin de un conjunto de actividades.
	Dentro del diagrama de proceso, se emplea para representar una actividad, si bien también puede llegar a representar un conjunto de actividades.
	Representa una decisión. Las salidas suelen tener al menos dos flechas (opciones).

	Representan el flujo de productos, información, y la secuencia en que se ejecutan las actividades.
	Representan a un conjunto de actividades o procesos

**Fuente.** Elaboración Propia

## 2.2 Sistema GPS

Al escuchar la palabra GPS lo primero que imaginamos es que el dispositivo que poseemos en la mano es la que proporciona toda la información para saber la ubicación exacta del lugar donde el dispositivo se encuentra.

En este apartado veremos algunos puntos relevantes para poder entender lo que es un sistema GPS y como está compuesto.

### 2.2.1 Tecnología GPS

GPS es un acrónimo de la expresión Inglesa (Global Positioning System) que en el español significa Sistemas de Posicionamiento Global. Es el primer sistema de navegación basada en satélites, implementado por los DoD, DoT y NASA (Departamentos de Defensa, Departamento de Transporte) de la agencia espacial norteamericana, como un sistema de navegación de precisión con fines militares.

El funcionamiento de sistema GPS consiste en la recepción de 3 y 8 señales de radio de satélites de los cuales se conoce de forma exacta su posición orbital con respecto a la tierra.

[1]

Para entender el sistema GPS se hace necesario saber 3 elementos esenciales por las que está conformada, estos segmentos se clasifican en: segmento espacial, segmento de control y el segmento de usuario.

## **2.2.2 Composición del sistema GPS**

### **2.2.2.1 Segmento espacial**

El segmento espacial está constituido por satélites que soportan el sistema y las señales de radio. Estos satélites son conocidos como la constelación NAVSTAR (acrónimo de la expresión inglesa Navigation Satellite Timing and Ranging), está constituido por 24 satélites operando y cuatro satélites de reserva.

La constelación NAVSTAR, permite que cualquier punto de la tierra pueda leerse simultáneamente entre 6 y 11 satélites, normalmente nominalmente SVs o Space Vehicles. Esto posibilita tener una observación durante las 24 horas del día. [2]

**Figura 2.2**  
**Constelación NAVSTAR**



**Fuente.** [Coreia, 2002]

### **2.2.2.2 Segmento de control**

El segmento de control es toda la infraestructura necesaria en la tierra para el control de la constelación NAVSTAR. Dicha constelación está controlada desde la tierra a través de una serie de cinco estaciones. Una de ellas está situada en Colorado Springs que es la estación principal nominada (Master Control), y las otras cuatro estaciones son nominadas (Monitor Station). Estas estaciones están colocadas estratégicamente cerca al

plano ecuatorial. Todas estas estaciones cuentan con receptores de relojes de muy alta precisión.

**Figura 2.3**  
**Estación de control a NAVSTAR**



**Fuente.** [1, fig. 2.5]

Los datos recogidos por las estaciones secundarias son enviados a la principal, donde son debidamente procesados, calculándose las efemérides<sup>2</sup>, el estado de los tiempos, etc. Toda esta información se transmite a los satélites en los cuales se almacena. [2]

#### 2.2.2.3 Segmento de usuario

El segmento de usuario está constituido por los instrumentos que el usuario necesita para la utilización del sistema GPS, es decir para la navegación, el cual le permitirá tener un posicionamiento y control de la precisión de tiempos. [2]

---

<sup>2</sup> Las Efemérides son Conjunto de parámetros numéricos que describen las posiciones precisas de los satélites en función del tiempo. Las mismas pueden ser transmitidas o precisas.

Uno de los instrumentos que el usuario necesita es un equipo GPS es decir un receptor o sensor con antena que pueda ser externa o integrada al censored, permitiendo así una comunicación con los satélites Esto permitirá, una interoperación entre el satélite y el usuario. Hoy en día, cada vez más se incluyen accesorios con diversas funcionalidades.

**Figura 2.4**  
**Segmento de usuario**



**Fuente.** Elaboración propia

### **2.2.3 Tipos de GPS**

Existen dos tipos de GPS que actualmente pueden adaptarse a las exigencias de los usuarios: Los Navegadores convencionales (como por ejemplo de GPS de bajo coste) y los Receptores de C/A<sup>3</sup> avanzados (como por Ejemplo de GPS más sofisticado).

---

<sup>3</sup> C/A Course/acquisition en español curso/adquisición.



### 2.2.3.1 Navegadores convencionales

Son los tipos de receptores GPS más extendidos, por el bajo costo y multiplicidad de aplicaciones. Consiste en receptores capaces de leer el código C/A capaces de leer señales diferenciales vía radio y capaces de representar en cartografías sencilla.

Permiten conocer las coordenadas en varios formatos y conversiones de baja precisión. También permiten la navegación con puntos de referencias, indicaciones asistidas, rumbos, direcciones y señales de llegada audibles de llegada en rutas definidas por el usuario a través de puntos de referencia (Waypoints)<sup>4</sup>.

Los precios de estos tipos de navegadores pueden ir desde los 200 a los 600 Euros aproximadamente y su precisión puede ir desde los 25 m a los 7 m. en planimetría<sup>5</sup> con una precisión de 16 metros de altimetría<sup>6</sup>. Dependiendo de la visibilidad de los satélites. Estos tipos de GPS no permiten trabajar con la base de datos geográficos definidos por el usuario ni permiten un almacenamiento de datos alfanuméricos personalizado. A cambio, presenta la gran ventaja de que usuario no tiene que tener ninguna formación específica para su manejo.

---

<sup>4</sup> Los *waypoints* son coordenadas para ubicar puntos de referencia tridimensionales utilizados en la navegación fundamentada en GPS.

<sup>5</sup> La *planimetría* consiste en proyectar sobre un plano horizontal los elementos de la poligonal como puntos.

<sup>6</sup> La *altimetría* se encarga de la medición de las diferencias de nivel o de elevación entre diferentes puntos del terreno.

**Figura 2.5**  
**Navegadores convencionales**



**Fuente.** Elaboración propia

#### 2.2.3.2 Receptores de C/A Avanzados.

Son receptores que además de analizar el código C/A disponen de lectura con ciertas limitaciones.

Estos receptores permiten el uso de metodologías diferenciales, en ocasiones bajo suscripciones a servicios vía satélite como OmniStar o LandStar, con una precisión de entorno de 1 m en tiempo real. Es posible llegar a esta precisión porque el DGPS<sup>7</sup> vía satélite permite hacer las correcciones en tiempo real.

Son muy aptos para los sistemas GIS<sup>8</sup> aparte de permitir una precisión compatible, permiten el manejo de la base de datos geográficas realizadas por el usuario.

---

<sup>7</sup> DGPS (*Differential Global Positioning System*) o *diferencial del sistema de posicionamiento global*

<sup>8</sup> GIS significa Sistema de Información Geografía

En cuanto a las correcciones diferenciales, es muy frecuente que esto sea vía satélite mediante suscripción a un sistema de pagos.

La suscripción suelen tener un precio anual de alrededor de 1.400 Euros para la precisión métrica de un país. Por otro lado se puede contratar el servicio por periodos limitados de semanas, meses o incluso días.

**Figura 2.6**  
**Receptores GPS C/A Avanzados**



**Fuente:** elaboración propia

#### 2.2.3.3 El método diferencial DGPS

Este método consiste en la utilización de un receptor móvil y una estación de referencia sobre coordenadas conocidas. La idea básica para comprender el fundamento del DGPS es la utilización de receptores sobre puntos de coordenadas muy bien conocidas, estos receptores conocidos como estaciones de referencia, leen en todo momento las posiciones reportadas por sus observaciones GPS y las comparan con las posiciones teóricas de sus coordenadas conocidas.

En tiempo real, las estaciones de referencia transmiten las correcciones a realizar a los receptores de los usuarios, que también están leyendo directamente la señal GPS y que al vuelo recoge dichas correcciones y las aplica a sus medidas con lo cual se mejora notablemente la precisión de los datos.

Por otro lado Lethan dice “Un DGPS convierte un receptor civil en tanto o más preciso que uno militar. Ahora que la disponibilidad selectiva ha sido eliminada y los receptores civiles ofrecen una precisión mínima de 15 metros, la mayor parte de las personas no necesitan DGPS. Sin embargo, El DGPS sigue siendo útil para una mayor precisión, aunque ello no compense el precio” [3].

Esto no significa que un receptor este ya haciendo correcciones diferenciales, sino que receptor el capaz de entender y aplicar la información correcta.

#### 2.2.3.4 Elección de método de receptor GPS

Como se ha visto existen diversos modelos de GPS que se pueden clasificar dependiendo del propósito para su uso, para la implementación del sistema GPS en el proyecto a desarrollarse, se debe tomar en cuenta cuál de estos tipos de GPS se adaptan en nuestro objetivo.

Haciendo un análisis el método que más se adapta a las necesidades del proyecto por su precisión y portabilidad de los datos, es el método (DGPS), que pertenece a los tipos de receptores sofisticados.

#### **2.2.4 Receptores GPS**

En el mercado se ha introducido una gran cantidad de dispositivos GPS. Cada año que pasa nuevos modelos son introducidos, sus tamaños oscilan desde grandes unidades que pesan varios kilos y que fueron diseñadas para embarcaciones o vehículos pesados, y pequeños receptores portátiles.

Los avances realizados mediante circuitos integrados han aportado una variedad de características entre los receptores. Por otra parte hay elementos comunes que todos los receptores poseen es la sensibilidad de las antenas, la precisión, el dato de la cuadrícula de los mapas. La importancia de las otras características es de como uno lo vaya a usar su receptor. Cada característica dependerá de lo que usted vaya a necesitar. [3, pp. 33]

Afortunadamente los receptores GPS actuales tienen mucha potencia y amplias funciones, por lo tanto no habrá problema de elegir un receptor que mejor se adapte para la implementación en el sistema.

#### 2.2.4.1 Funcionalidad del GPS

Los receptores GPS tienen un elemento esencial Para la recepción de datos enviados por los satélites, este elemento es la Antena GPS. Lethan resalta algo muy importante sobre la antena GPS en la que dice “Poco importará el peso de su receptor, cuanto duran la baterías, cuantos canales tiene o de que otras prestaciones dispone, si su antena no puede detectar las señales del satélite en tal caso el receptor no podrá calcular su posición. Hay nace la frase tanto pagas, tanto tienes” [3, pp. 34].

Una vez que el receptor tiene los datos enviados por los satélites, el receptor envía los datos vía GSM, SMS o GPRS al cliente, dependiendo que canal está habilitado para él envío o salida de datos.

#### 2.2.5 Sistema GPRS

GPRS (General Packet Radio Service) que en español significa Servicio General de Paquete vía Radio, está diseñado netamente para la transmisión de datos mediante paquetes desde terminales móviles, creada en la década de los 80 es una extensión del Sistema Global para la Comunicación Móviles (Global System for Mobile Communications o GSM).

Para entender más la relación o diferencia entre tecnología GPRS y GSM veremos algunos puntos relevantes que nos ofrecen esta tecnología.

#### 2.2.5.1 GSM

Los sistemas GSM es conocida como es sistema de comunicación de 2ª generación basadas en células de radio, apareció como respuesta a los problemas de los sistemas de comunicación analógicos.

Las redes GSM tienen ciertas limitaciones para la transmisión de datos de las cuales se mencionan a continuación:

- La velocidad de transferencia es de 9,6 Kbps.
- Tiempo de establecimiento de conexión, de 15 a 30 segundos.
- Pago por tiempo de conexión.
- Problemas para mantener la conectividad en itinerancia (Roaming).

Como se mencionó, las características que ofrece esta tecnología para la transmisión de datos es de 9,6 Kbps, con esta velocidad el usuario no puede navegar por internet de manera satisfactoria. Además debemos tomar en cuenta que el pago se realiza por el tiempo de conexión.

Los factores que limitan a la tecnología GSM permiten a esta tecnología que solo sea mayormente para utilizada para la voz (llamadas) y no para datos. [4]

#### 2.2.5.2 GPRS

La tecnología GPRS es la solución a las tecnologías GSM para el envío de datos, a través de terminales móviles, esta tecnología unifica el mundo IP con el mundo de las telefonías móvil y a través de ella creando toda una red GSM orientada exclusivamente para la transmisión de datos. Al sistema GPRS se lo conoce también como GSM-IP ya que usa la tecnología IP (Internet Protocol) para acceder directamente a los proveedores de contenidos de internet.

Se puede resumir en tres Características importantes sobre las bondades que ofrece esta tecnología para el envío de datos:

- Velocidad de transferencia desde los 44 Kbps hasta los 144 Kbps.

- Conexión permanente. Tiempo de establecimiento de conexión inferior a un segundo.
- Pago por la cantidad de información transmitida, no por tiempo de conexión.

Como se pudo ver entre las características que no ofrecen esta tecnología es que un usuario GPRS puede estar conectado todo el tiempo que desee ya que no se necesita mucho recurso porque mientras no envíe o reciba datos no paga.

La tarificación se lo realiza por el volumen de datos en lugar de tiempo de uso.

La velocidad de transferencia de datos es mayor a los de GSM. [4]

### 2.2.6 Dispositivos GPS

Como se ha mencionado en el apartado de la sección 2.1.4, existen una gran variedad de dispositivos GPS en el mercado.

En la siguiente tabla se describirán algunos modelos de receptores GPS y sus características.

**Tabla 2.2**  
**Tabla de comparación de receptores**

MODELO RECEPTOR	DESCRIPCIÓN
<b>MVT600</b> 	<ul style="list-style-type: none"> <li>• Envía de datos por 3 canales GPS, GPRS, SMS.</li> <li>• Ranura para memoria externa tarjeta micro SD.</li> <li>• Micrófono (opcional).</li> <li>• Botón de Alarma SOS</li> <li>• Cámara (Opcional).</li> <li>• Alarma de impacto.</li> <li>• Precio 250\$us.</li> <li>• Duración de batería interna (12-72) horas.</li> </ul>

**MVT340**

- Envía de datos por 3 canales GPS, GPRS, SMS.
- No existe Ranura para memoria externa tarjeta micro SD.
- Botón de alarma SOS.
- No Micrófono.
- No Cámara.
- Alarma de impacto.
- Precio 150\$us

**TK 103**

- Envía de datos por 3 canales GPS, GPRS, SMS.
- No existe Ranura para memoria externa tarjeta micro SD.
- Botón de alarma SOS.
- No Micrófono.
- No Cámara.
- No Alarma de impacto.
- Precio 170\$us

**TK 106**

- Envía de datos por 3 canales GPS, GPRS, SMS.
- existe Ranura para memoria externa tarjeta micro SD.
- Botón de alarma SOS.
- Micrófono (opcional).
- No Cámara.
- Alarma de impacto.
- Precio 200\$us
- Duración de batería interna (48-120) horas

**Fuente.** Elaboración propia



Como se puede observar en la Tabla 2.2 se identifican los modelos de receptores cada uno con sus respectivas descripciones que hacen que un receptor sea distinto al otro.

#### 2.2.6.1 Elección del receptor GPS.

Para la implementación del dispositivo en el sistema a desarrollarse se tomó en cuenta que el receptor MVT340 brinda las funcionalidades básicas y comunes que brindan una gran mayoría de receptores que se encuentran a la venta en nuestro medio.

Se decide utilizar el modelo de receptor MVT340 por las siguientes razones.

- Con costo económico a diferencia de los demás receptores.
- Cumple con los funcionamientos básicos en el envío de datos de la TRAZA<sup>9</sup>.
- Sistema interno manipulable para la configuración del direccionamiento de envío de datos a un servidor.
- Es uno de los modelos más conocidos en nuestro medio.

Pero sin embargo para la implementación en la Asociación de Transporte mixto 24 de septiembre se recomienda utilizar el modelo MVT600 por las siguientes razones.

Cuenta con un circuito para la implementación de micrófono con una entrada estéreo, esto ayudaría al socio al momento de comunicarse con el chofer o inversa.

Con implantación de una cámara en el receptor el chofer podrá enviar imágenes de lugares o almacenar en la memoria externa.

#### 2.2.6.2 Funciones del receptor MVT340.

A continuación se mencionaran algunas funciones relevantes que brinda el receptor MVT340.

---

<sup>9</sup> TRAZA son datos representados en línea de carácter en la cual están los datos tales como fecha, hora, latitud, longitud, velocidad, temperatura, etc

- Modulo GPS SiRF III, Modulo GSM de cuatro bandas de funcionamiento 850/900/1800/1900Mhz.
- AGPS (con la estación base GSM).
- Seguimiento por SMS / GPRS (TCP / UDP) (Protocolo de MEITRACK).
- Monitoreo en demanda.
- Monitoreo por medio de intervalos de tiempo.
- Monitoreo a través de intervalo de distancia.
- Monitoreo por medio de equipos móviles (celulares).
- Memoria interna de 4 MB de almacenamiento.
- Sensor de movimiento incorporado (Tremble sensor).
- Batería interna 850mAh.
- Alarma de SOS.
- Alarmas de Geo-cerca.
- Alarma GPS de fuera de cobertura.
- Alarma de batería baja.
- Alarma de exceso de velocidad.
- Alarma de Impacto.
- Alarma de remolque.
- Alarma de Corte de Antena GPS.
- Alarma de corte De alimentación externa.
- Reporte de Kilometraje.
- Inmovilización de vehículo.
- Magneto interno (opcional).
- 2 Entradas digitales( 1 de Activación positiva, 1 de activación negativa).
- 1 control de salida (Para inmovilización de vehículo).

#### 2.2.6.3 Accesorios de un receptor

Los equipos de rastreo MVT340 contienen los siguientes accesorios incluido en su paquete original de fábrica así como se muestra en la figura 2.7.

**Figura 2.7**  
**Accesorios de un Receptor**



**Fuente.** [5, Fig. 1]

1. Cables de entrada y salida.
2. Cable de datos USB.
3. CD, este contiene aplicaciones y manuales que sirve de guía para configurar el rastreador.
4. MVT340 unidad principal con batería interna (incluida).
5. Antena GPS.
6. Antena GSM.

#### 2.2.6.4 Características del receptor MVT340.

**Tabla 2.3**  
**Características de un receptor MVT340**

Descripción	Especificación
Dimensión	110x72x39mm
Peso	170g

Tensión de carga	DC 9V~36V/1.5mA
Batería de repuesto	850mAh/3.7V
Temperatura de operación	-20°C~55°C
Humedad	5%~95%
Horas de funcionamiento	43 horas en modo de ahorro de energía y 10 horas en modo normal
LED	2 luces LED para mostrar GPS, GSM y otros estados
Botón	Un SOS y un encendido / apagado
Micrófono	Ninguno
Memoria	Ninguno
Sensor	Sensor de movimiento
Frecuencia SM	GSM 850/900/1800/1900MHz
Modulo GPS	Ultimo GPS SiRF Star III chipset-
Sensibilidad PS	-159dB
Precisión de posicionamiento	10 metros, 2D RMS
Entradas y salidas E/S	2 entradas digitales (1 negativa y 1 positiva) 1 entrada análoga 1 salida 1 puerto USB para configuración por medio de PC únicamente.

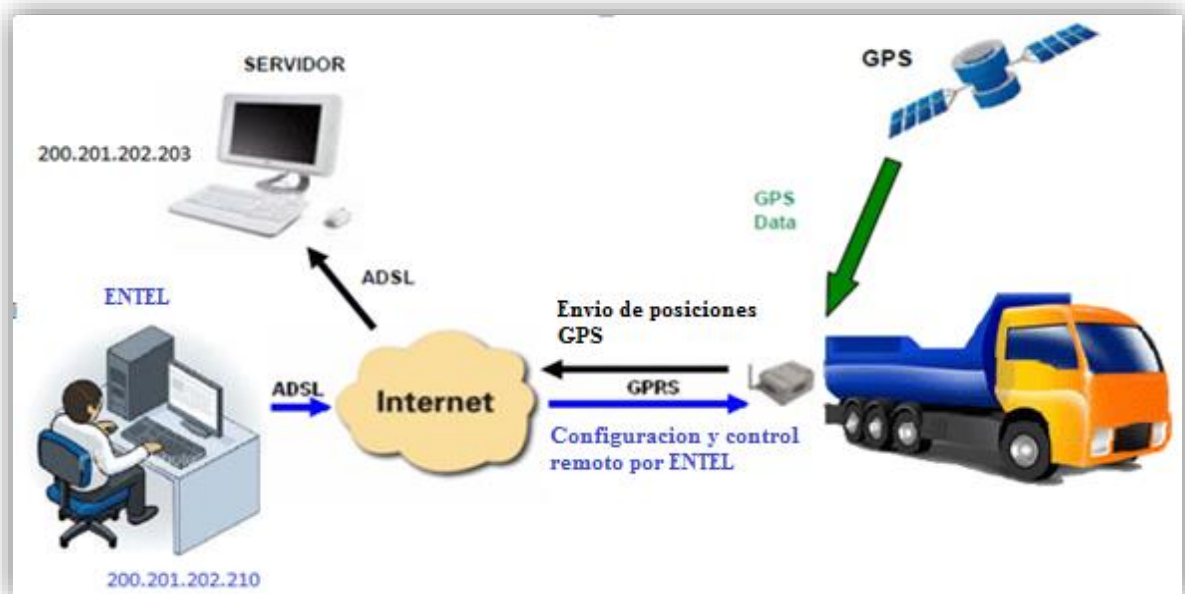
**Fuente.** [5, Pg. 3.]

### 2.3 Proceso de envío de datos de un receptor

Los receptores GPS para el proceso de envío de datos o un traza necesitan tener una configuración interna en donde se especifica la dirección o la IP<sup>10</sup> del servidor donde estará alojado para el almacenamiento de los datos que el receptor envía.

En la siguiente figura se muestra el esquema general del proceso de envío de datos a un servidor.

**Figura 2.8**  
**Envío de datos de un receptor al servidor**



**Fuente.** Elaboración propia

Como se muestra en la figura superior el esquema general del proceso de envío de datos a un servidor debemos considerar los siguientes requisitos para lograr la implementación del dicho modulo en el sistema.

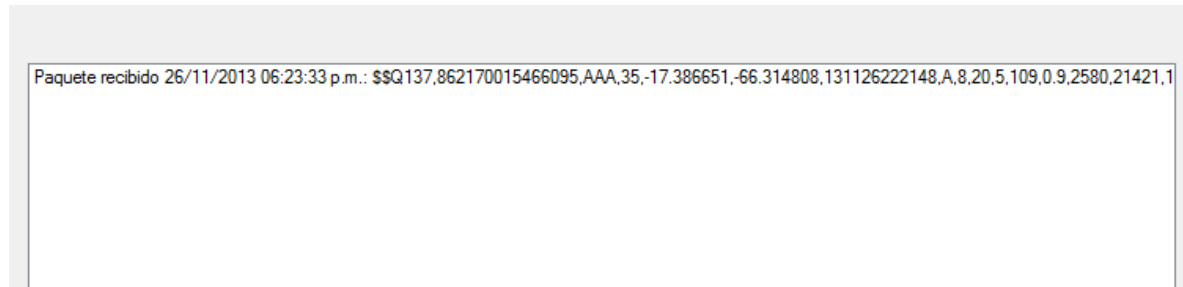
- Tener una IP fija.

<sup>10</sup> Las direcciones IP (IP es un acrónimo para Internet Protocol) son un número único e irrepetible con el cual se identifica una computadora conectada a una red que corre el protocolo IP.

- Configurar nuestro GPS para el envío de los datos (TRAZA) a esta IP y un puerto determinado en nuestro servidor, Por ejemplo puerto: 5625.
- Crear un programa que capture la TRAZA en el servidor y permita almacenar en la base de dato para esto se desarrollara un SOCKET.
- Ahora, configuramos nuestro receptor GPS para que envíe la información a la IP fija que tiene el servidor y al puerto 5625 esta pueden ser TCP<sup>11</sup> o UDP<sup>12</sup>.

En la siguiente figura se muestra el proceso de llegada de datos des dispositivo al servidor.

**Figura 2.9**  
**Captura de una traza del receptor GPS**



**Fuente.** Elaboración propia

### 2.3.1 Definición y características de un Socket

Los sockets son mecanismos de comunicación entre procesos que permiten que un proceso hable, emita reciba información con otro proceso incluso estando en distintas máquinas. Una forma de conseguir que dos programas se transmitan datos. [18]

Desde un punto de vista de programación, un socket no es nada más que un “fichero” que se abre de una manera especial.

<sup>11</sup> TCP, *Protocolo de Control de Transmisión*) El protocolo TCP garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

<sup>12</sup> UDP, son siglas de ( *Protocolo de Datagrama de Usuario*) un protocolo si conexión que, como TCP, funciona en redes mediante IP

### 2.3.2 Propiedades de un Socket

Las propiedades que los socket brindan en la transmisión de datos son los siguientes:

- Fiabilidad de la transmisión, no se pierden los datos transmitidos.
- Conservación del orden de los datos, los datos llegan en el orden en el que se enviaron.
- No duplicación, los datos solo llegan una vez
- Comunicación en modo conectado, la conexión está establecida antes de iniciar la comunicación, de este modo la emisión desde un extremo va destinada al otro.
- Conservación de los límites de los mensajes. Los límites de los mensajes emitidos pueden encontrarse o conectarse en el destino.
- Envío de mensajes urgentes.

### 2.3.3 Atributos de un Socket

Los socket se caracterizan generalmente por tres atributos:

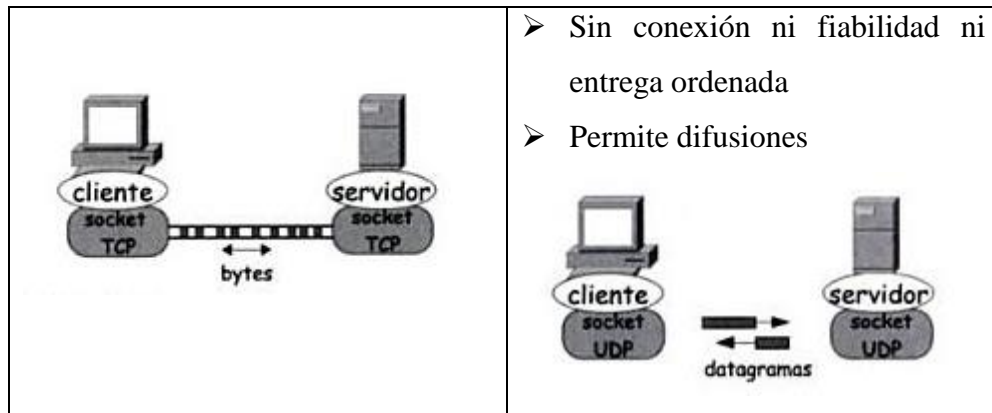
- Dominio, especifica el medio de comunicación de la red que el socket utilizará.
- Protocolo, especifica que protocolo se va a usar.
- Tipo, los protocolos de internet proveen dos niveles distintos de servicio: flujo y datagramas.

### 2.3.4 Tipos de Socket

En la siguiente tabla describiremos los tipos de sockets que existen.

**Tabla 2.4**  
**Tipos de sockets**

SOKETS TCP	SOKETS UDP
<ul style="list-style-type: none"><li>➤ Las aplicaciones piden al S.O. una comunicación controlada por TCP.</li><li>➤ Orientada a la conexión</li><li>➤ Comunicación fiable y ordenada</li></ul>	<ul style="list-style-type: none"><li>➤ Las aplicaciones piden al S.O. una comunicación controlada por UDP.</li><li>➤ Transferencia de bloques de datos</li></ul>



**Fuente.** [23, Figura 3.2]

## 2.4 Ingeniería de Software

La ingeniería de software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producto de software. Por otro lado Pressman lo define de la siguiente manera “La Ingeniería de Software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales”. [6]

### 2.4.1 Procesos de desarrollo de software clásicos

Estos modelos son aquellos que requieren hacer un minucioso análisis de requerimientos antes del proceso de implementación, además de regirse estrictamente a normas o reglas extremas. Entre ellas se mencionaran a continuación a algunas modelos.

- Modelo de la Espiral.
- Modelo de la cascada.
- Modelos de proceso incremental.
- Modelo de proceso evolutivo.



### **2.4.2 Métodos de desarrollo ágil**

Las características de los modelos ágiles pueden explicarse a través de los siguientes principios que son esenciales entenderlos, para comprender la filosofía que se maneja:

- La prioridad más alta es satisfacer al cliente a través de la entrega pronta y continua de software valioso.
- Son bienvenidos los requerimientos cambiantes, una en una etapa avanzada del desarrollo.
- Entregar con frecuencia software que funcione, de dos semanas a un par de meses.
- El cliente y la interacción es más importante que los procesos y las herramientas.
- Es más importante que el software funcione, que tener una documentación extensa.
- La negociación de contratos es inferior a la colaboración con el cliente.
- Respuesta ante cambios inesperados, es más importante que el seguimiento de un plan.

A continuación se mencionaran algunas metodologías de proceso de desarrollo de software ágiles.

#### **2.4.2.1 Scrum.**

Scrum es una metodología en que se aplica de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras.

Es definida también como un marco para la gestión de proyectos, es uno de los más populares en el uso e información.

Esta metodología es muy apropiada para proyectos que requieren cambios rápidos de requisitos.

Sus principales características se las puede resumir en dos. El primero es que el desarrollo de software se realiza mediante iteraciones, denominadas Sprint con una duración de un mes natural y hasta de dos semanas si se necesita. El resultado de cada sprint tiene que proporcionar un resultado completo, un incremento del producto final.

La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas podemos destacar las reuniones diarias de 15 minutos, estas reuniones son para coordinación e integración en la que cada miembro del equipo debe responder a tres preguntas:

- ¿Qué he hecho desde la última reunión?
- ¿Qué voy a hacer desde este momento?
- ¿Qué impedimento tengo o voy a tener? [6, cap.3, pp. 68-70].

#### 2.4.2.2 Programación Extrema XP.

Es una metodología ágil propuesta por Kent Beck, centrada en potenciar las retroalimentaciones con el cliente como clave para el éxito en desarrollo del software, promoviendo el trabajo en equipo entre los programadores, preocupándose por el aprendizaje de los mismos desarrolladores y propiciando un buen clima de trabajo. XP se basa en:

- Realimentación continua entre el cliente y el equipo de desarrollo.
- Comunicación fluida entre todos los participantes.
- Simplicidad en las soluciones implementadas.
- Coraje para enfrentar los cambios.

La programación Extrema es especialmente adecuada para proyectos con requisitos imprecisos no bien definidos, muy cambiantes, y donde existe un alto riesgo técnico.

#### 2.4.2.3 Proceso Unificado Ágil (PUA).

El proceso unificado ágil (PUA) adopta una filosofía “en serie para lo grande” e “iterativa para lo pequeño”. Al adoptar las actividades clásicas del PU.

El PUA brinda un revestimiento en serie (por ejemplo, una serie lineal de actividades de ingeniería de software) esto permite que el equipo visualiza el funcionamiento general de un proyecto de software.

En cada actividad el equipo repite los procesos de elaboración con el objetivo de alcanzar la agilidad para entregar lo más rápido posible un incremento de software significativo a los usuarios finales.

Cada iteración del PUA está compuesta por las siguientes actividades que se mencionaran:

- Modelado. Se crean representaciones UML de los dominios de negocio e el problema, estos modelos deben ser muy buenos y bien definidos esto con el fin de que el equipo avance.
- Implementación. Los modelos se definen en código.
- Pruebas. Al igual que con XP el equipo diseña una serie de pruebas y las ejecuta esto con el fin de garantizar que el código fuente cumple con sus requisitos.
- Despliegue. En esta actividad se centra en la entrega de un incremento de software para tener una retroalimentación de los usuarios finales.
- Configuración y administración del proyecto. En este proceso se debe realizar un plan para la administración de cambios y el riesgo.
- Administración del ambiente. La administración debe coordinar para disponer toda la infraestructura que se va a necesitar. [6, pp. 74-76]

**Tabla 2.5**

**Cuadro de comparación de las metodologías ágiles y tradicionales**

<b>METODOLOGÍAS ÁGILES</b>	<b>METODOLOGÍAS TRADICIONALES</b>
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios
Impuestas internamente (por equipo).	Impuestas externamente.
Procesos menos controlados, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos	Existe un contrato prefijado.

es bastante flexible.	
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes posiblemente distribuidos.
Pocos artefactos	Más artefactos.
Pocos roles	Más roles.
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelo

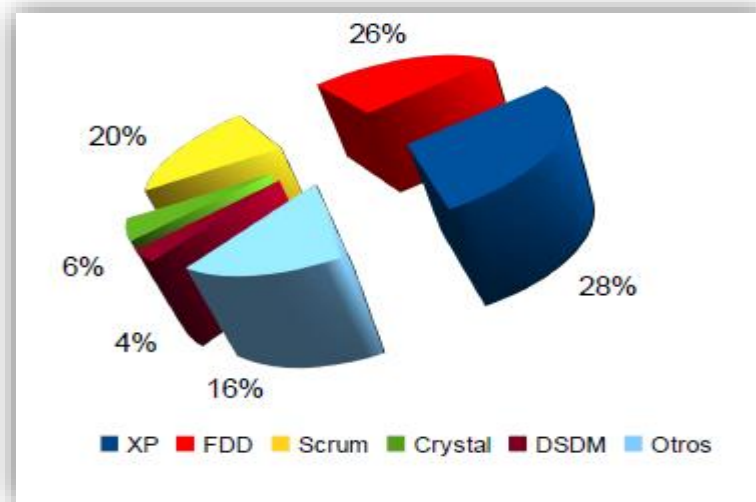
**Fuente.** [7, Tabla1.1, pg.12.]

#### 2.4.2.4 Elección de la metodología de desarrollo

Se elige la metodología del proceso de desarrollo XP por las características que se mencionaron y la flexibilidad que nos ofrece dicha metodología a esto se suma la gran cantidad de información disponible que existe.

Se elige la metodología de proceso de desarrollo XP por sus características y por la existencia de la mayor información acerca de esta metodología. Además de que se tiene una gran simpatía por este modelo, según la encuesta que hizo CM Crossroads entre los lectores de Configuration Management Journal en Agile Journal, de los modelos más usados, ver gráfico 2.10

**Grafica 2.10**  
**Preferencia de modelos ágiles**



**Fuente.** Elaboración Propia

“es XP la metodología que resalta por contar con la mayor cantidad de información disponible y es con diferencia la más popular” (6 Sec. 3.1).

También por consiguiente a continuación la lista de las razones del por qué se elige esta Metodología “Programación Extrema”.

- Tiene como fundador a un reconocido personaje de entre los que fundaron el manifiesto Ágil, Kent Beck.
- Existen libros de experiencias prácticas, teoría sobre XP.
- Grandes desarrolladores del software libre utilizan XP como metodología.
- Mucha información acerca de XP en español.
- Fama entre los lectores de Configuration Management Journal.

#### **2.4.3 Programación Extrema.**

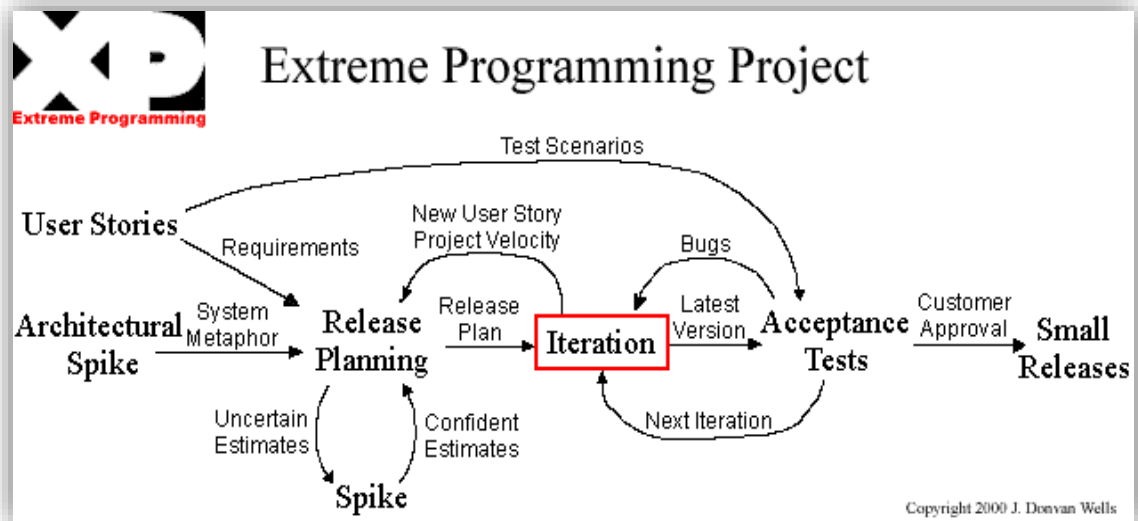
Kent en uno de sus discursos defines de la siguiente manera “XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software”.

#### 2.4.3.1. Faces de vida de un proyecto con XP.

El ciclo de vida de XP consiste de seis fases.

**Figura 2.11**

#### **Fases de un proyecto en Programación Extrema**



**Fuente.** [6 Fig. 3.5]

#### 2.4.3.2 Exploración.

En esta fase, en que el cliente plantea a grandes rasgos las historias de usuario.

Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. [6, Sec. 3.4].

#### 2.4.3.3 Planificación de la entrega.

En esta fase el cliente establece la prioridad de cada historia de usuario, y los programadores realizan una estimación del esfuerzo necesario para implementar cada una de ellas.

Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. [6, Sec. 3.4].

#### 2.4.3.4 Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado.

El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fueren la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. [6, Sec. 3.4].

#### 2.4.3.5 Producción

Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento). [6, Sec. 3.4].

#### 1.4.3.6 Mantenimiento

Esta fase es un poco difícil de hacerlo con personal muy reducido.

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura. [6, Sec. 3.4].

#### 2.4.3.6 Muerte del proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo. [6, Sec. 3.4]

### 2.4.4 Lenguaje Unificado de Modelado (UML)

Es el lenguaje de modelado de sistema de información más conocido y utilizado en la actividad. Es un lenguaje basado en diagramas para visualizar, especificar, construir y documentar un sistema, además de contar con una gama de herramienta de diseño.

“Cuando esté considerando usar el UML, es importante preguntarse porque lo hará y como le ayudara a usted cuando llegue el momento de escribir el código. No existe una evidencia empírica adecuada que demuestre si estas técnicas son buenas o malas”. [11]

#### 2.4.4.1 Diseño UML en XP

Una vez elegida la metodología XP para el desarrollo de software se sigue la secuencia que tiene sobre el lenguaje de modelado UML de esto habla Martin Fowler en su sitio web.



Estoy seguro de que a pesar de la impresión extrema, XP no es solo prueba, codificar y refactora. Hay espacio para el diseño antes de codificar.

Selectividad es la clave para usar bien el UML. No dibujes cada clase solo las importantes.

Para cada clase no muestres cada atributo y operación solo las importantes. No dibujes diagrama de secuencia para todos los casos de uso y escenario, solo los importantes.

Un problema común en el uso de los diagramas es que la gente intenta hacerlos extensos. El código es la mayor fuente de información extensa ya que el código es lo más fácil de mantener en sincronía con el código. [12]

Los diagramas son útiles en la medida en que ayuda al desarrollo del proyecto por esa razón es que seleccionamos unas cuantas que se aplicaran solo cuando sean necesarias.

UML al paso del tiempo va mejorando su estructura, añadiendo nuevos diagramas, y versiones.

Teniendo un conjunto de diagramas solo de mencionaremos las más significativas y explicativas que se usaran en el proyecto.

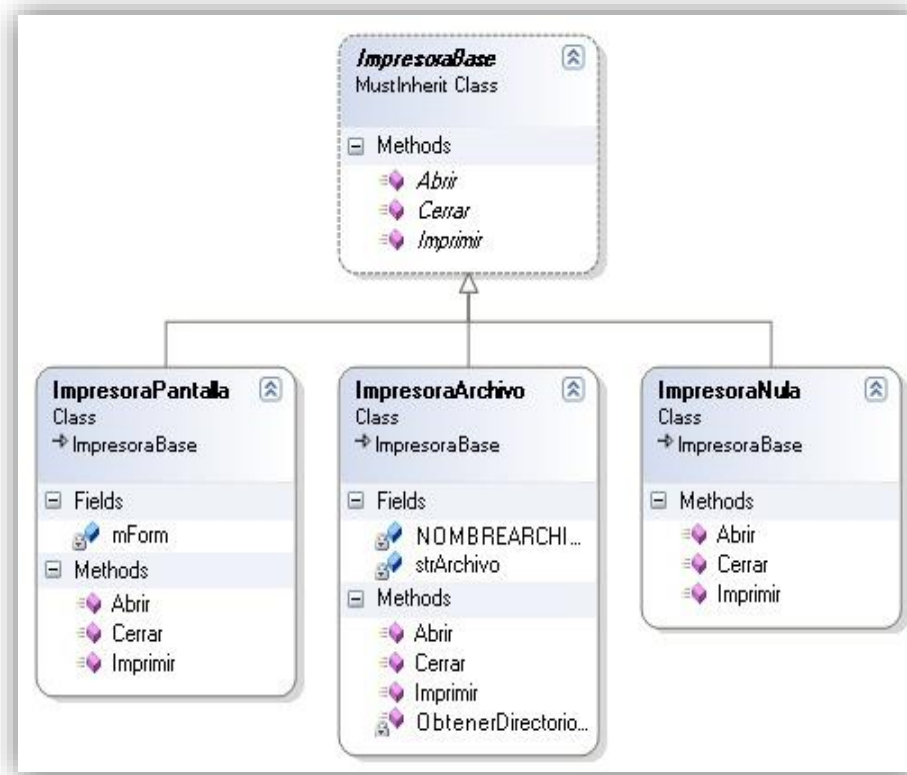
#### 2.4.4.2 Diagrama de Clases.

Un diagrama de clases sirve para visualizar las relaciones entre las clases que invocan el sistema, las cuales pueden ser asociativas, de herencia, de uso y de consentimiento. [10 cap. 8, Pg.210]

Un diagrama de clases está compuesto por los siguientes elementos:

- **Clase:** Atributos, métodos y visibilidad.
- **Relaciones:** Herencia, composición, Agregación y uso

**Figura 2.12**  
**Diagrama de Clases**



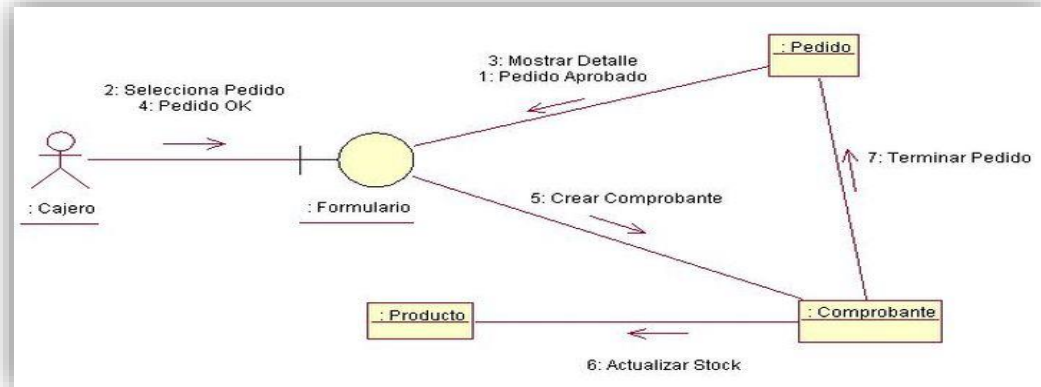
**Fuente.** [11 Fig. 2.5]

#### 2.4.4.3 Diagrama de Iteración

UML incluye los llamados diagramas de interacción que sirven para mostrar ejemplos de cómo ciertos objetivos interactúan a través de mensajes para la realización de tareas. Existen varios tipos de diagramas de interacción que son equivalentes entre sí pero en particular estos son los que más se destacan.

- Diagramas de secuencias.
- Diagramas de colaboración.

**Figura 2.13**  
**Diagrama de colaboración.**



**Fuente.** [26]

#### **2.4.5 Enterprise Architect como herramienta para UML**

Enterprise Architect provee modelo de ciclo de vida completo para ingeniería de software y sistemas con capacidad de gestión de requisitos.

Enterprise Architect es una herramienta gráfica multiusuario diseñada para ayudar al equipo de desarrollo a construir sistemas robustos y mantenibles.

A continuación se describen algunas de sus características de Enterprise Architect los cuales apoyarán en el presente proyecto.

Trazabilidad de punta a punta. Enterprise Architect provee trazabilidad completa de desde los modelos de requisitos, análisis y diseño hasta la implementación y despliegue.


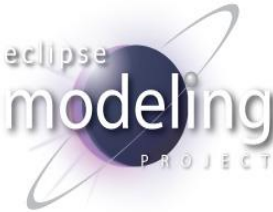
La verificación, validación efectiva y el análisis del impacto inmediato son posibles a través del ciclo de vida completo, usando capacidades tales como la matriz de relaciones y la vista de jerarquías de Enterprise Architect.



Generación e ingeniería inversa de código fuente. Enterprise Architect soporta la generación e ingeniería inversa de código fuente para muchos lenguajes populares incluyendo a C, C++, Java, Visual Basic, C#, PHP, Delphi.

Construido sobre UML. Usa los perfiles UML para extender el dominio de modelado, la validación del modelado asegura la integridad.

En la actualidad existen diferentes herramientas que te ayudan a modelar UML, pero cada uno tiene un diferente enfoque, por lo consiguiente se mencionaran algunas herramientas con sus respectivas características.

**Tabla 2.6**  
**Tabla de comparación de herramientas UML**

Herramientas	Características
<b>Enterprise Architect</b> 	<ul style="list-style-type: none"> <li>• Es una herramienta para el análisis y diseño UML cubriendo el desarrollo de software desde la información, análisis, diseño, pruebas y mantenimiento.</li> <li>• Windows.</li> <li>• Usada por analistas, evaluadores, administradores de proyectos, personas de control de calidad, equipo de desarrollo.</li> <li>• Los 13 diagramas de UML 2.0</li> </ul>
<b>Eclipse Plugin UML 2.0</b> 	<ul style="list-style-type: none"> <li>• Es una herramienta para desarrollar software en entorno integrados.</li> <li>• Multiplataforma</li> <li>• Desarrollo aplicaciones de manera profesional es extensamente utilizado en actividades didácticas.</li> <li>• Diagramas de actividad, clases, componentes, compuesto, despliegue, caso de uso.</li> </ul>

<p><b>ArgoUML</b></p> 	<ul style="list-style-type: none"> <li>• ArgoUML es una aplicación de diagramado de UML Escrita en Java y publicado bajo Licencia BSD.</li> <li>• Multiplataforma.</li> <li>• Sus características se ajustan a proyectos académicos y de investigación.</li> <li>• Diagrama de actividad, clase, colaboración, despliegue, secuencia, caso de uso.</li> </ul>
<p><b>Software Ideas Modeler</b></p> 	<ul style="list-style-type: none"> <li>• Permite crear fácilmente diagramas UML con solo arrastrar los elementos.</li> <li>• Multiplataforma.</li> <li>• Proyectos académicos y comerciales.</li> <li>• Todo los diagramas UML.</li> </ul>

**Fuente.** Elaboración propia

Las herramientas mencionadas anteriormente poseen valiosas características de UML que ayudan no solo a la elaboración del modelo de desarrollo de software, sino también a diferentes áreas de la ingeniería.

Se ve conveniente el uso de la herramienta Enterprise Architect, por la necesidad de modelar diferentes diagramas y a esto se suma el conocimiento de la herramienta en anteriores proyectos.

## **2.5 Patrón de arquitectura de software**

Los patrones de arquitectura de software son aquellos que expresan un esquema organizativo estructural lo cual es muy fundamental para un sistema de software.

Según Weitzendfeld [13] “La arquitectura de software define una estructura general de un sistema”.

### **2.5.1 Patrón de diseño Modelo Vista Controlador**

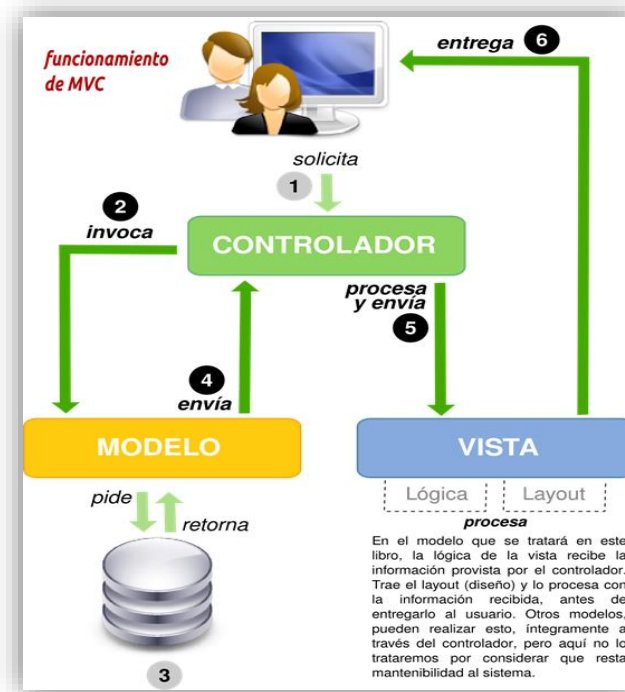
Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC<sup>13</sup> se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista. [14]

Al estar separando la lógica de interfaz de usuario y desacoplando el acceso a datos y otras llamadas desde la vista, la interfaz de usuario (aspx) puede mantenerse a sí mismo mientras que la lógica y el acceso a datos están dentro de la aplicación. [14]

---

<sup>13</sup> Modelo vista controlador

**Figura 2.14**  
**Diagrama de la arquitectura patrón MVC**



**Fuente:** Elaboración propia

- **Modelo.** El modelo representa la información con la que trabaja la aplicación es decir su lógica de negocio.

El modelo encapsula la capa de acceso a dato. Logrando así que una llamada a la capa de acceso a datos no se la pueda realizar si no es por medio de la capa de negocio.

- **Vista.** La vista transforma el modelo en una página web que permite al usuario interactuar con el mismo, por lo tanto la vista no es más que la interface donde el usuario interactúa a través de los eventos con el controlador y también para poder observar los resultados.

- **Controlador.** El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelado o en la vista.

El controlador usa el modelo para conseguir los datos o almacenarlos y el controlador vuelve a usar las vistas para comunicar los resultados, que el usuario solicita.

### **2.5.2 MVC con la Arquitectura N-Capas**

La arquitectura Modelo vista controlador puede ser usado con la arquitectura N-Capas, la cual consiste en:

Las capas se ocupan de la división lógica de componentes y funcionalidad, y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lenguajes.

Las capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o el servicio. Esto les ayudara diferenciar entre los diferentes tipos de tareas a ser realizadas por los componentes, ofreciendo un diseño que maximiza la reutilización y especialmente, la mantenibilidad.

Cada capa lógica de primer nivel puede tener un número concreto de componentes agrupados en sub-capas. Dichas sub-capas realizan a su vez un tipo específico de tareas.  
[15]

### **2.5.3 Arquitectura N-Capas**

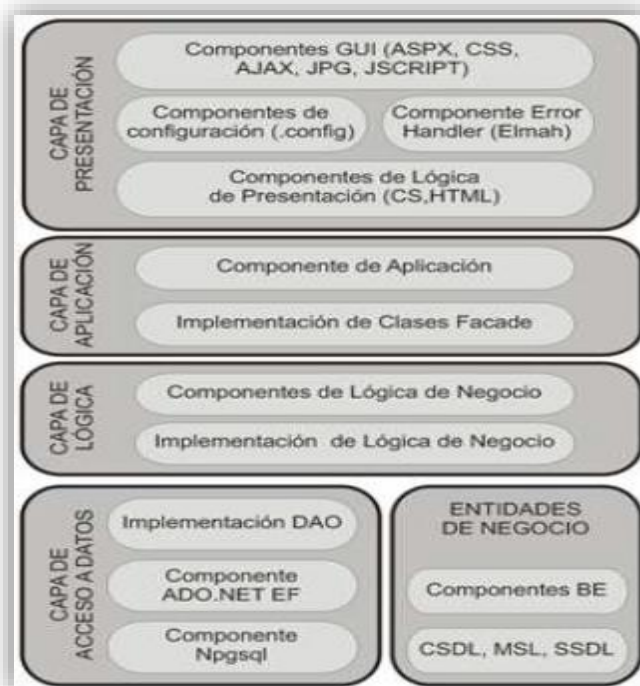
“Lo que se conoce como arquitectura en capas es en realidad un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento, etc.”.[15]

La arquitectura N-Capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan.



Para poder entender cómo interactúa una capa con la otra, LLORENTE explica mediante el siguiente cuadro.

**Figura 2.15**  
**Arquitectura N-Capas**



**Fuente.** [15 fig. 3.4]

Como se puede observar en la figura superior describiremos el flujo de trabajo de la arquitectura dividida en cuatro capas.

A continuación se describirán las funcionalidades y operaciones que se realizan en cada capa.

#### 2.5.3.1 La capa de presentación

Esta capa integra los elementos de la interfaz gráfica y las clases con la lógica del comportamiento de las páginas para su interacción con el usuario. Involucra librerías CSS, JavaScript, Ajax, Flash, páginas maestras y ficheros ASPX y HTML además de contenido audiovisual. Esta capa actúa de forma similar a la Vista en el patrón MVC.

#### 2.5.3.2 Capa de Aplicación

Esta capa tiene como función delegar las solicitudes de usuario provenientes de la capa previa hacia los módulos y clases correspondientes de la Capa de Lógica de Negocio, sin involucrar la implementación en líneas de código de dicha solicitud. Asimismo actúa como fachada para futuras implementaciones de integración con otros dispositivos, plataformas y sistemas a través de aplicaciones como servicios Web.

#### 2.5.3.3 Capa de Lógica

Conformada por clases cuyas funciones recaen en la implementación de la lógica de negocio tendiendo el requerimiento de usuario. Interactúa con la capa de base de datos de acuerdo con el tratamiento deseado de la información intercambiada.

#### 2.5.3.4 Capa de Acceso a Datos

En esta capa se ubicarán las clases DAO y librerías de conexión encargadas de administrar las operaciones CRUD (Create – Read – Update – Delete) y sentencias SQL a nivel de base de datos.

Para el desarrollo del proyecto se utilizara arquitectura N-Layer por las siguientes ventajas que mencionaran a continuación según LLORENTE menciona [15, pp. 38].

Flexibilidad. Permite que los componentes sean modificados para llevar a cabo sus tareas sin necesidad de recompilar toda la aplicación.

Mantenibilidad. Facilita la tarea de modificar un componente para corregir errores, mejorar el desempeño, agregar atributos, o adaptarlos a un ambiente cambiante.

Reutilización. Todos los componentes pueden ser utilizados desde otros componentes o desde otros sistemas.

Escalabilidad. Facilita que un componente se pueda adaptar al cambio. Cuando el sistema crece en funcionalidad pero esta está definida por diferentes clientes.

## **2.6 Herramientas de Programación**

Las herramientas de programación permiten realizar programas, rutinas y sistemas para que la parte física de un computador u ordenador, funciones y pueda producir resultados.

### **2.6.1 Plataforma .NET**

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones, que nos brinda todas las herramientas y servicios que se necesitan para el desarrollar aplicaciones empresariales y de misión crítica. “Microsoft .NET representa el cambio hacia un modelo de programación en el que los dispositivos, servicios y ordenadores trabajan juntos para proporcionar soluciones a los usuarios”.

Componentes de la plataforma:

- Entorno de Ejecución (Runtime)
- Bibliotecas de funcionalidad(Class Library)
- Lenguajes de programación
- Compiladores
- Herramientas de desarrollo(IDE & Tools)
- Guías de arquitectura

Es importante mencionar que es importante escribir código en la plataforma de desarrollo Microsoft.NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para para ser usado en ella, C# carece de elementos heredados innecesarios en .NET por esta razón se puede decir es Lenguaje nativo de .NET [18]

Además “Microsoft diseño C# de modo que retuviera toda la sintaxis de C y C++” [18]. Este lenguaje es orientado a objetos moderno y seguro.

Algunas de sus características no son propias del lenguaje sino de la plataforma .NET son las siguientes:

- Sencillez.

- Modernidad.
- Orientado a objetos.
- Orientado a componentes.
- Gestión automática de memoria.
- Seguridad de tipos.
- Instrucciones seguras.
- Sistema de tipo unificado.
- Extensibilidad de tipos básicos.
- Extensibilidad de operadores.
- Extensibilidad de modificadores.
- Eficiente.

#### 2.6.1.1 Microsoft Visual Studio 2012

“El .NET Framework incluye todas las herramientas necesarias para desarrollar aplicaciones .NET. El código fuente se puede escribir con cualquier editor de texto simple, como por ejemplo el editor de notas de Windows, y se compila con los compiladores que proporciona el .NET Framework SDK<sup>14</sup> [19].”

Según Jhonson [19, pp. 19] lo primero que resalta a la vista al abrir la aplicación es la nueva perspectiva. Lo cual la interfaz ha sido completamente rediseñada para simplificar el flujo de trabajo y dar fácil acceso a todas las herramientas. Se han simplificado las barras de herramientas y todo es mucho más accesible, facilitando, de esta manera, la navegación por toda la aplicación.

De tal forma Visual Studio 2012 es totalmente compatible con la nueva versión de Windows, ofrece nuevas plantillas, pantallas de diseño, herramientas de evaluación y depuración, con las que crear aplicaciones en el menor tiempo posible, aplicaciones Windows Phone 8, Windows Azure, Sharepoint, desarrollo de Juegos y Aplicaciones 3D.

---

<sup>14</sup> SDK Software Development Kit

#### 2.6.1.2 ASP .NET MVC4

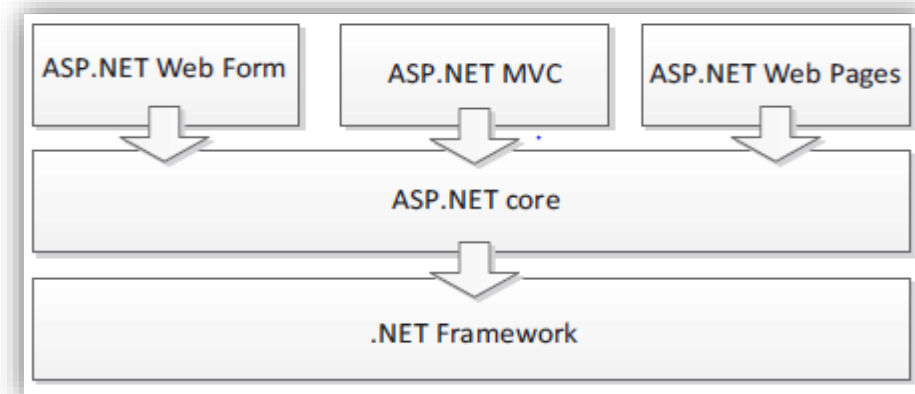
Microsoft ASP.NET MVC es un framework de desarrollo de aplicaciones Web. Se apoya en gran medida en los patrones de desarrollos probados y prácticas que ponen énfasis en una arquitectura con código altamente mantenible.

La última evolución de la plataforma ASP.NET de Microsoft proporciona un modelo de programación con alto nivel de productividad y promueve una arquitectura de código más limpia, orientada al desarrollo y con poderosa extensibilidad en combinación con todos los beneficios de ASP.NET

MVC significa Modelo-Vista-Controlador, un patrón de diseño que es muy popular en el espacio de desarrollo Web. Como alternativa a los formularios Web Forms, ASP.NET MVC tiene un enfoque diferente cuando se trata de la estructuración de las aplicaciones Web.

Tanto ASP.NET Web Forms y ASP.NET MVC se sientan uno al lado del otro en la parte superior de la plataforma central ASP.NET, como se muestra en la figura 2.15

**Figura 2.16**  
**Relación entre varias tecnologías Web ASP .NET**



**Fuente:** [20, Fig.1.2]

MVC 4 viene con muchas mejoras y nuevas características, además de la nueva dependencia. NET 4. Estas nuevas características incluyen las siguientes características:

- El motor de vistas Razor.
- Gestión de paquetes con NuGet.
- Mejora la extensibilidad.
- Filtros de Acción Global.
- Funciones de lenguaje dinámico.
- Parcial caché de resultados de página Mejoras.
- Ajax.
- Plantillas Móviles.
- API Web.

Junto con MVC 4 ya viene incorporado HTML5 el cual se utilizara para poder crear las aplicaciones multiplataforma en la Web. Otra ventaja son los Web Api que proporciona MVC 4, esta característica ayudara con la comunicación con la Aplicación móvil consumiendo Rest Full y Json logrando el intercambio de datos entre ellos. Cabe mencionar que la integración de la herramienta a utilizar es la adecuada y se acomoda bien con los requisitos del sistema.

#### 2.6.1.3 Razor: Motor de Vista para ASP.Net

“ASP.NET MVC siempre ha apoyado el concepto de “motores de vistas”, donde las vistas realizan tareas solo de presentación [14].

Según Shepherd [14, pp. 296] algunas de las características de Razor son:

- La minimización de las características y pulsaciones de teclado en un archivo.
- Utiliza todos los lenguajes existentes y conocimientos de HTML
- Permite utilizar los lenguajes de programación a la preferencia del desarrollador.
- Requiere una herramienta específica y le permite ser productivo en cualquier editor de texto.
- Está apoyada con la capacidad de realizar pruebas unitarias sobre las vistas.

## 2.7 Gestores de Base de Datos

Las bases de datos actuales son todas relacionales con un conjunto de tablas que se encuentra dividido en filas y columnas. Para acceder a las bases de datos relacionales, IBM<sup>15</sup> desarrollo el lenguaje SQL el cual fue implementado con el pasar del tiempo en todos los motores de base de datos.

### 2.7.1 Microsoft SQL Server 2008 Express

SQL Server es una plataforma para base de datos que se usa en el procedimiento transaccional en línea (OLTP) a gran escala en las bodegas de datos y las aplicaciones de comercio electrónico así como también es una plataforma de inteligencia de negocios para soluciones de integración, análisis y creación de informes de datos.[21]

**Figura 2.17**  
**Plataforma de datos SQL Server**



**Fuente.** [MICROSOFT]

Una base de datos SQL Server está dividida en varios componentes lógicos, como tablas, vistas y otros elementos que son visibles al usuario.

---

<sup>15</sup> IBM. International Business Machines, una de las compañías de fabricación de ordenadores en el mundo desde los años 50 fue la creadora del PC, también se la conoce como el gigante azul.

Un servidor SQL Server puede contener varias bases de datos pertenecientes a diversos usuarios. Cuando se crea una base de datos, a sus usuarios se les puede asignar permisos de acceso, eso posibilita que varias bases de datos sean almacenados por SQL Server y que el acceso a cada una de ellas sea limitado a usuarios específicos.

### **2.7.2 Elección de Gestor de Base de Datos**

SQL Server 2008 Express es el servidor adecuado para trabajar con Visual Studio 2012 .NET Framework y Entity Framework basado en el enfoque Code Firsts siendo una de las etapas en el desarrollo del software. Lo aconsejable a la hora de diseñar un sistema de información completa, es procurar en la medida de lo posible no mezclar tecnologías de diferentes fabricantes.

### **2.7.3 Entity Framework: Code First**

*Code First* es un enfoque más de Entity Framework (hay otros dos enfoques que son Database First y Model First) que plantea lo siguiente: “Tú crea clases POCO (Plain Old CLR Objects) con tu lenguaje favorito (C#, VB.NET, etc.) y crea relaciones entre las mismas, después despreocúpate que ya veré yo como me las gasto para persistir tu modelo en una base de datos” [22].

Lo importante es entender que con Code First, lo primero es el código. En vez de comenzar creando la base de datos y después con ingeniería inversa generar las clases POCO. Con Code First primero creamos el modelo con código y después se genera automáticamente la base de datos. Lo cierto es que Code First también puede trabajar con base de datos existentes.

Code First reporta las siguientes ventajas:

- En el proyecto sólo se habla de código, no se habla más de bases de datos ni de T-SQL, al fin y al cabo, la base de datos será simplemente un forma más de persistir en el (la base de datos será un medio, no el fin).
- Parece que, definitivamente, se elimina el desajuste de impedancia. Esto es que el proyecto ya no vive en dos mundos, el de la base de datos y el del código, ya no se tiene la



necesidad de saber T-SQL además de C#, ya sólo con nuestras clases POCO y Linq se puede abordar cualquier proyecto.

Cierto es que estas ventajas también se las consigue con Database First y Model First, porque al fin y al cabo un modelo es siempre el mismo con independencia del enfoque del que provenga. Sin embargo, con Code First todo parece más fluido y con la opción de que sólo se puede abrir la consola de administración de Sql Server para confirmar que el modelo ha persistido y que no es todo una ilusión y que los datos están en el limbo.

En conclusión Code First nos acerca una opción mucho más cercana al código, que nos provee:

- Diseñar una base de datos sin la necesidad de un diseñador.
- Definir nuestro modelo a partir de entidades POCO.
- Utilizar las convenciones definidas por defecto para no especificar ningún tipo de configuración.
- Opcionalmente especificar una configuración a través de una Fluent API provista por el propio framework.

Es por estas bondades ya mencionadas y además tomando en cuenta que Code First trabaja muy bien con MVC4 se opta por utilizar este modelo.

## **2.8 Seguridad**

La seguridad informática Areitio lo define de la siguiente manera “la seguridad informática es una disciplina en continua evolución. La meta final de la seguridad es permitir que una organización cumpla con todos sus objetivos de negocio o misión, implementando sistemas que tengan un especial cuidado y consideración hacia los riesgos relativos a las TIC<sup>16</sup> de la organización, a sus socios comerciales, clientes, administración pública, suministradores”. [24]

---

<sup>16</sup> TIC, Tecnología de Información y la Comunicación

La seguridad informática debe garantizar la “confidencialidad, integridad, disponibilidad y autenticidad”.

### **2.8.1 Confidencialidad**

Es el requisito que intenta que la información privada o secreta no se revele a individuos no autorizados. La protección de la confidencialidad se aplica a los datos almacenados durante el procesamiento, mientras se transmiten y se encuentran en el tránsito.

### **2.8.2 Integridad**

Se encarga de garantizar que la información del sistema no haya sido alterada por usuarios no autorizados evitando la pérdida de consistencia. La cual se presenta en dos fases.

La primera fase es la Integridad de datos, consiste en que los datos no hayan sido alterados de forma no autorizada, mientras se almacenan, procesan o transmiten.

La segunda consiste en la Integridad del sistema, consiste en la cualidad que posee un sistema cuando realiza la función deseada, de manera no determinada y libre de manipulación. [24, pp. 11]

### **2.8.3 Disponibilidad**

Consiste en tener acceso a la información sin interferencia y sin obstrucción. Disponibilidad no implica que la información sea accesible para cualquier usuario, lo que significa es que tenga disponibilidad solo para usuarios autorizados.

El sistema presenta información oportuna y organizada en tiempo de ejecución, especialmente en el seguimiento y control de tareas, permitiendo ver los diferentes estados de una tarea asignada.

### **2.8.4 Autenticidad**

Es la “capacidad de verificar que un usuario convenientemente identificado, que accede a un sistema o genera un documento es quien dice ser”. [24, pp. 12]

## **2.9 Calidad de Software**

La calidad de software es un conjunto de cualidades que lo caracterizan determinando su eficiencia y utilidad satisfaciendo las necesidades tanto implícita como explícita del cliente. La IEEE en el estándar ISO 610-1990 lo define como el grado con que un sistema, componente o proceso cumple con los requisitos especificados y las necesidades o expectativas del cliente o usuario. [25]

A continuación se mencionan algunas normalizaciones para la calidad de software

- Documentación.
- Herramientas CASE.
- Evaluación de productos software y métricas.
- Gestión de ciclo de vida.
- Evaluación de procesos.
- Gestión de calidad.
- Ciclo de vida para la PYME<sup>17</sup>.
- Gestión de servicios.
- Pruebas.
- Arquitectura.
- ISO/TC159/SC4, formatos comunes para la usabilidad.

Como se puede observar en la lista mencionada en la parte superior existen varias normalizaciones para definir la calidad del software, sin embargo en el sistema a desarrollarse no se implementaran todas las normalizaciones.

## **2.10 Pruebas Unitarias**

Es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

---

<sup>17</sup> PYME, La Pequeña y Mediana Empresa

La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas [25].

### **2.10.1 Unit test**

Existen muchas definiciones acerca de lo que realmente es una prueba unitaria unit test sin embargo algunas son ambiguas y otras tienen un enfoque determinado de acuerdo a la situación en que se utilizan o se desarrollan.

Un unit test es un método que prueba una unidad de código. Al hablar de una unidad de código nos referimos a un requerimiento. Muchos desarrolladores tienen su propio concepto de lo que es una prueba unitaria; sin embargo, la gran mayoría coincide en que una prueba unitaria tiene las siguientes características:

- Prueba solamente pequeñas cantidades de código. Solamente prueba el código del requerimiento específico.
- Se aísla de otro código y de otros desarrolladores. El unit test prueba exclusivamente el código relacionado con el requerimiento y no interfiere con el trabajo hecho por otros desarrolladores.
- Solamente se prueban los endpoints públicos. Esto principalmente porque los disparadores de los métodos privados son métodos públicos por lo tanto se abarca el código de los métodos privados dentro de las pruebas.
- Los resultados son automatizados. Cuando ejecutamos las pruebas lo podemos hacer de forma individual o de forma grupal. Estas pruebas las hace el motor de prueba y los resultados de los mismos deben de ser precisos con respecto a cada prueba unitaria desarrollada
- Repetible y predecible: No importa el orden y las veces que se repita la prueba, el resultado siempre debe de ser el mismo.

- Son rápidos de desarrollar. Contrariamente a lo que piensan los desarrolladores que el desarrollo de pruebas unitarias quita tiempo los unit test por lo general deben de ser simples y rápidos de desarrollar. Difícilmente una prueba unitaria deba de tomar más de cinco minutos en su desarrollo.

## **CAPÍTULO 3**

### **MARCO PRÁCTICO**

En el siguiente capítulo se realizará el análisis físico, lógico de software, dando énfasis en la descripción de los procesos y funcionalidad del sistema.

Utilizando para el desarrollo del sistema la metodología de Programación Extrema “XP”. Lo cual, el cliente se encargara de plantear historias de usuario, de tal manera que se obtendrá una idea clara de sus necesidades y así realizar una primera entrega.

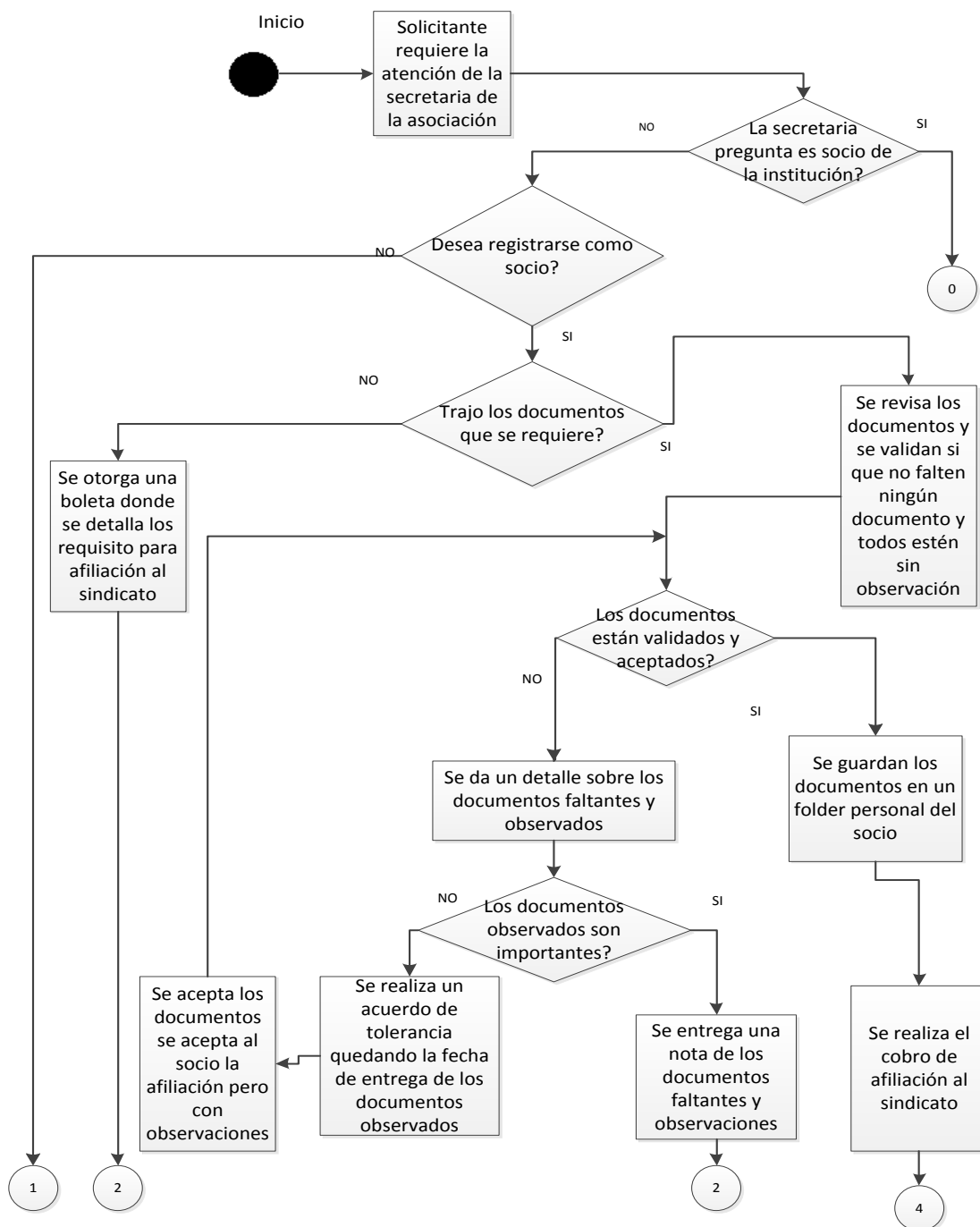
#### **3.1 Modelado de negocio**

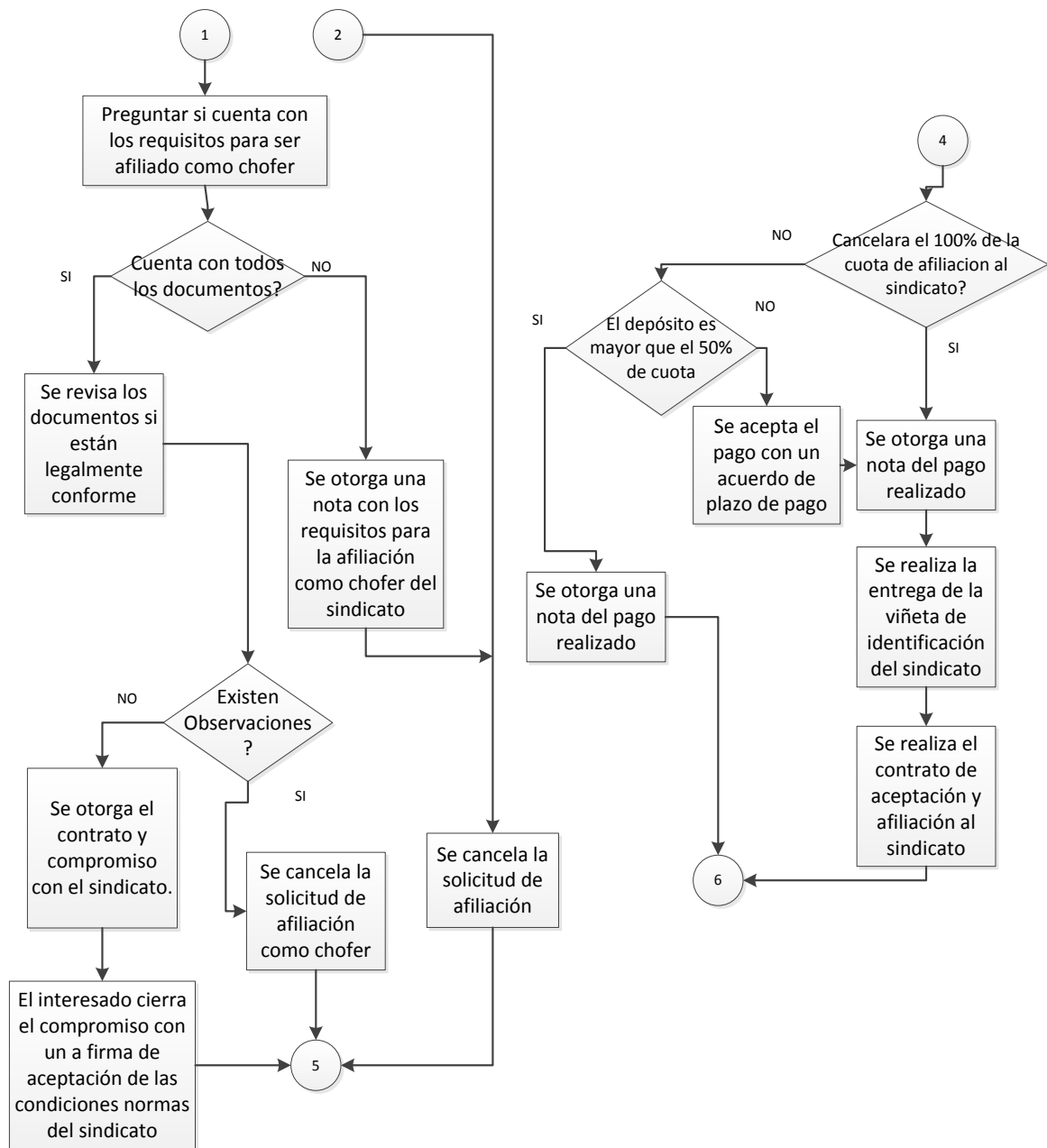
Presenta los procesos de negocios actual y alternativo que presentan el proceso que se realiza en la Asociación de Transporte Mixto 24 de Septiembre.

##### **3.1.1. Modelado de Negocio Actual**

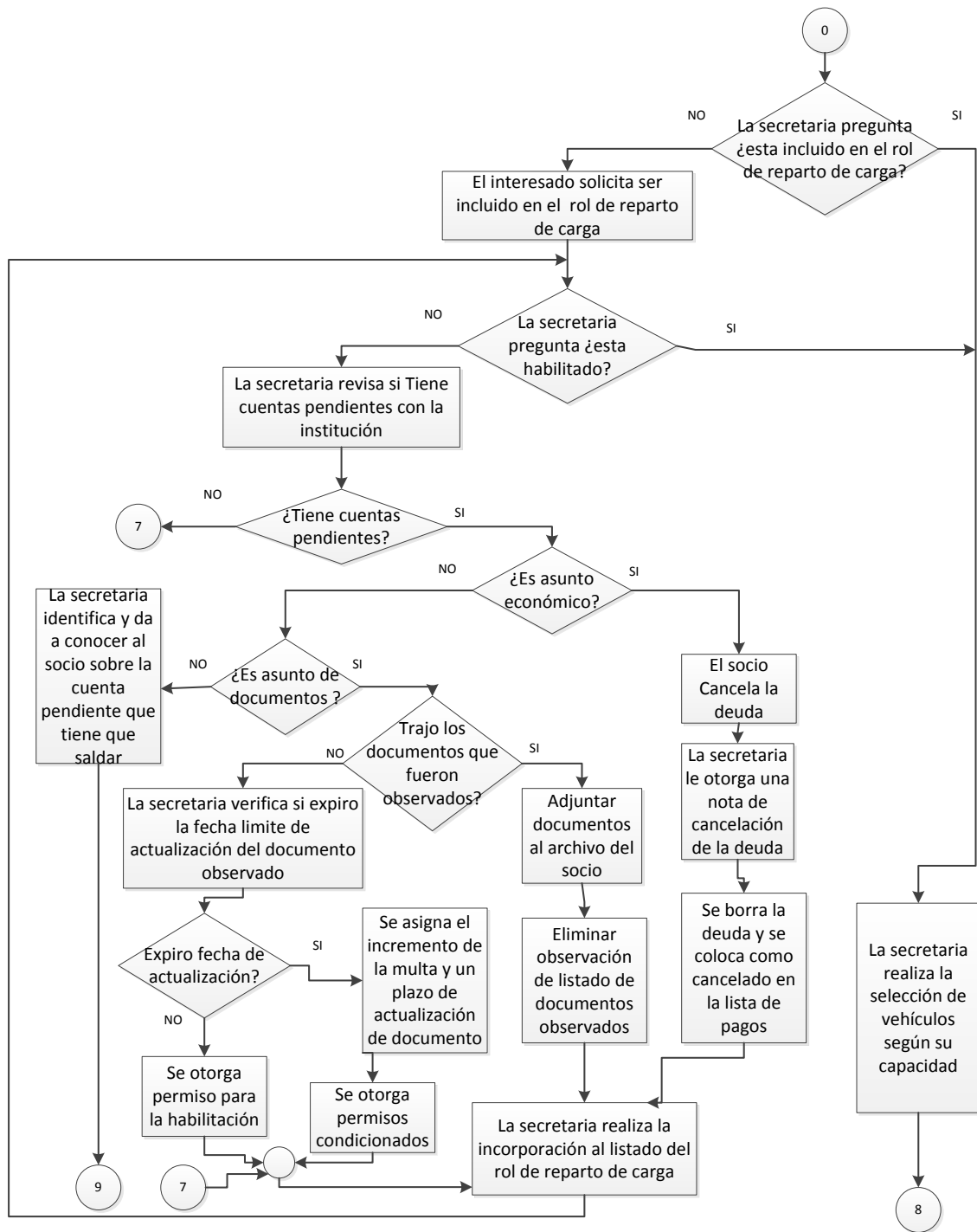
Como se puede observar en la siguiente figura, se muestra los procesos que se realizan actualmente en la Asociación de Transporte Mixto 24 de Septiembre en el proceso de planificación de envío de carga y monitoreo de los vehículos.

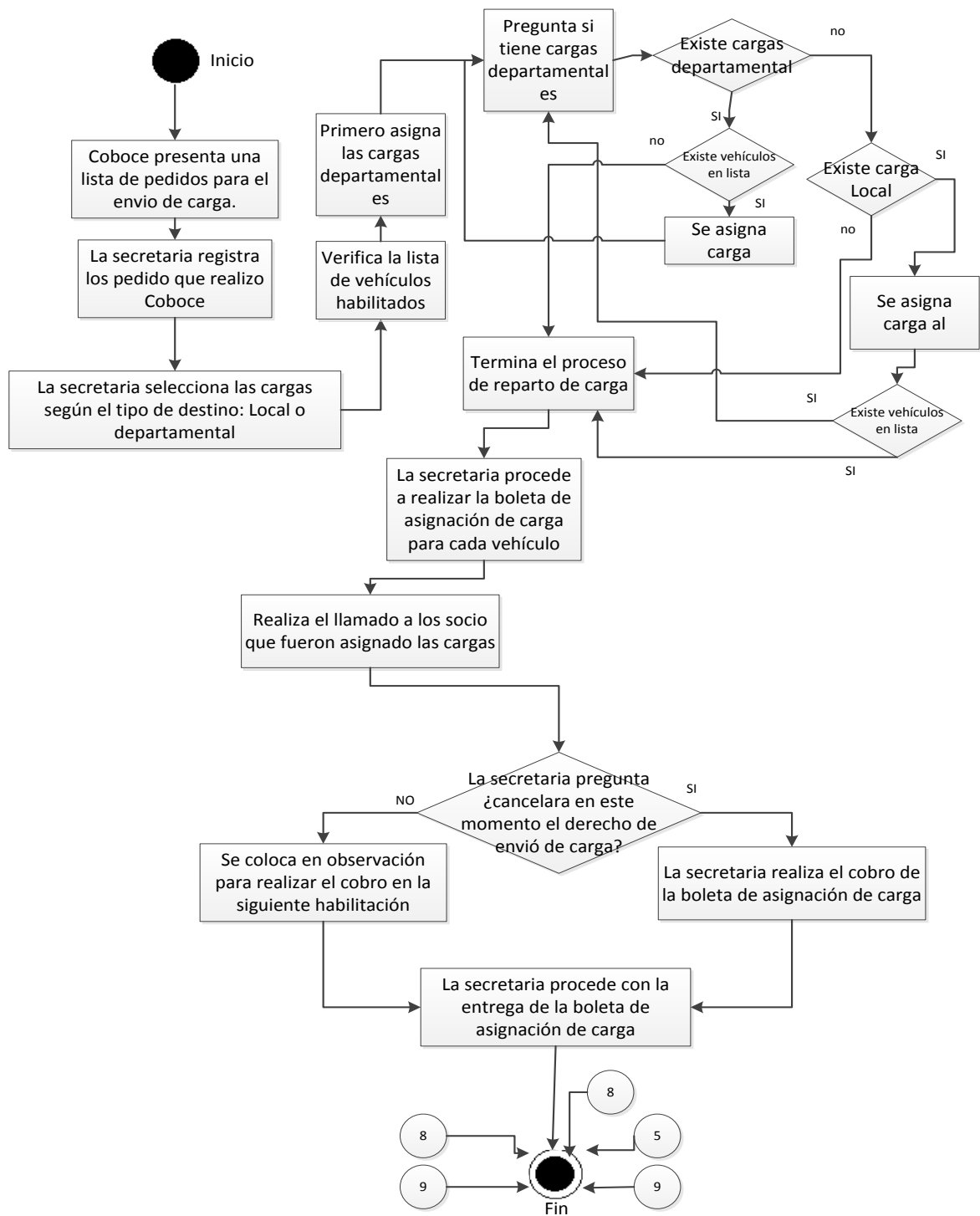
**Figura 3.1**  
**Modelado de negocio actual**









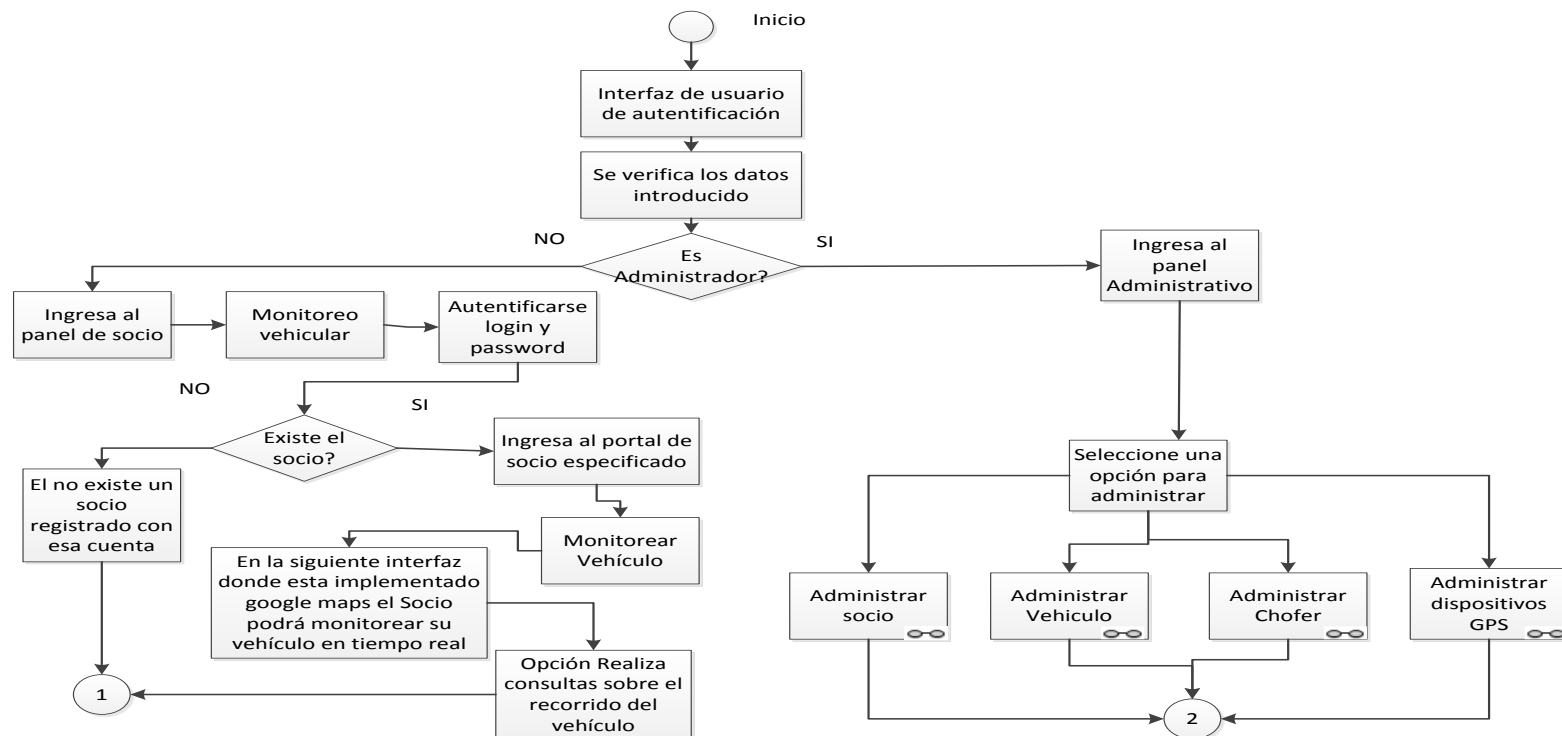


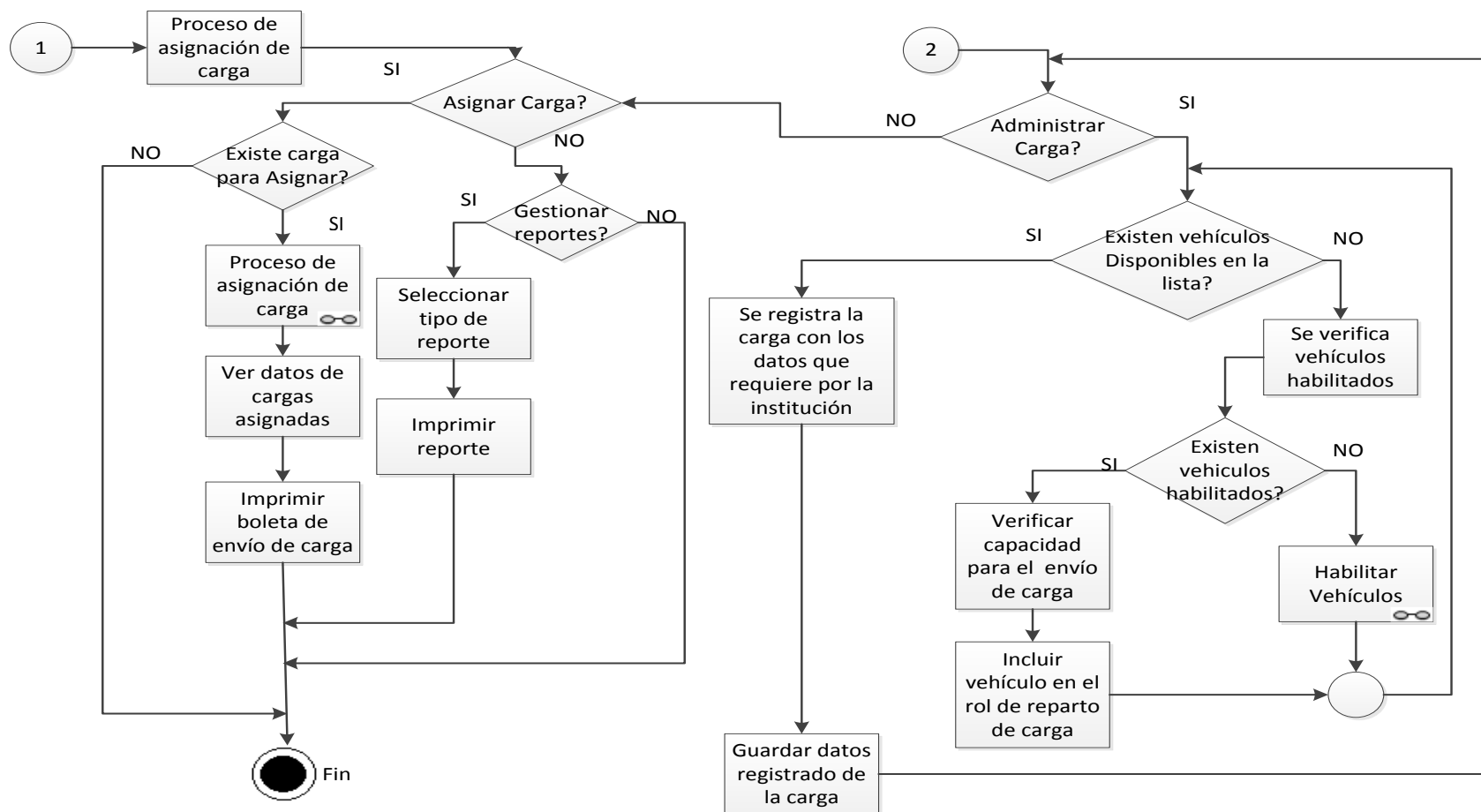
**Fuente.** Elaboración propia

### 3.1.2 Modelado de negocio alternativo

En la figura 3.5 se puede visualizar el nuevo proceso propuesto para la asociación de transporte mixto 24 de septiembre.

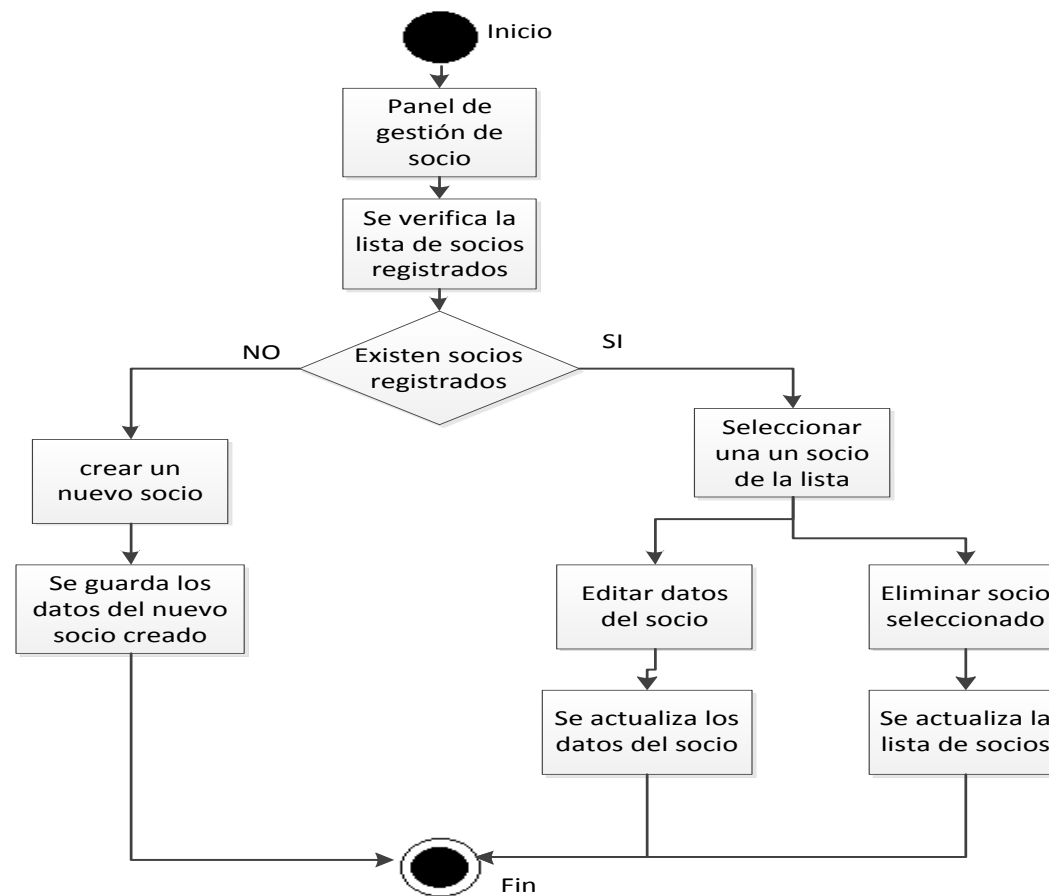
**Figura 3.2**  
**Modelado de negocio alternativo**





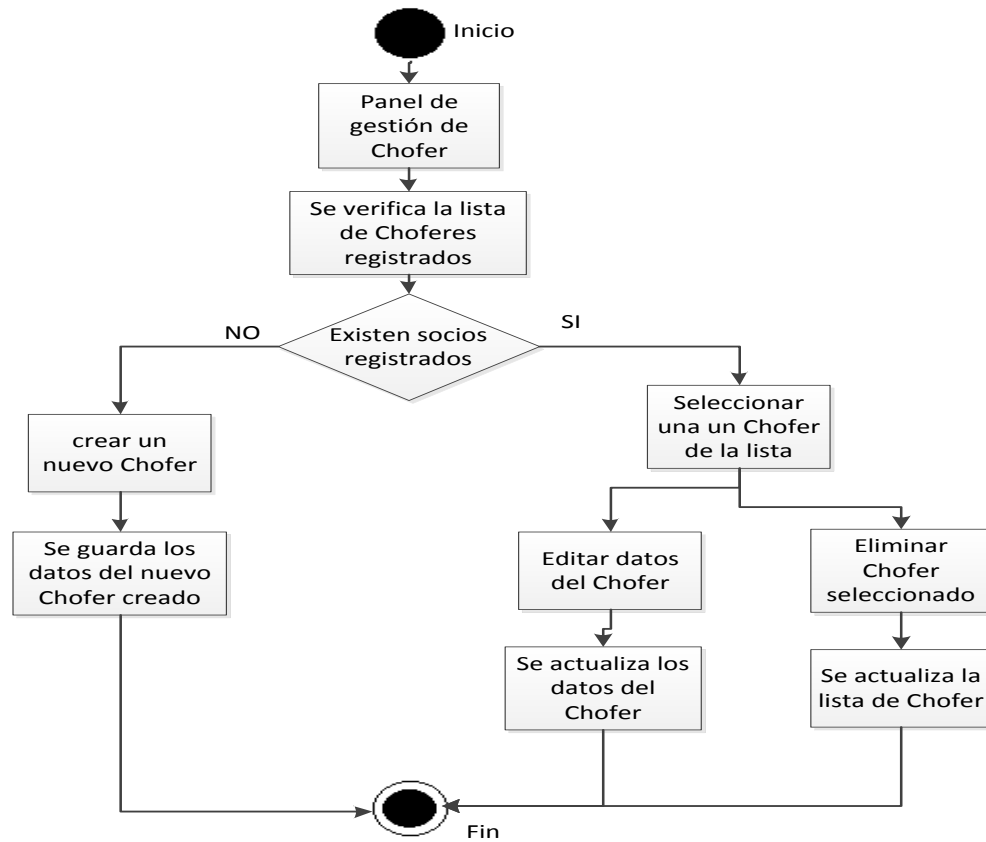
**Fuente.** Elaboración propia

**Figura 3.3**  
**Gestionar socio**



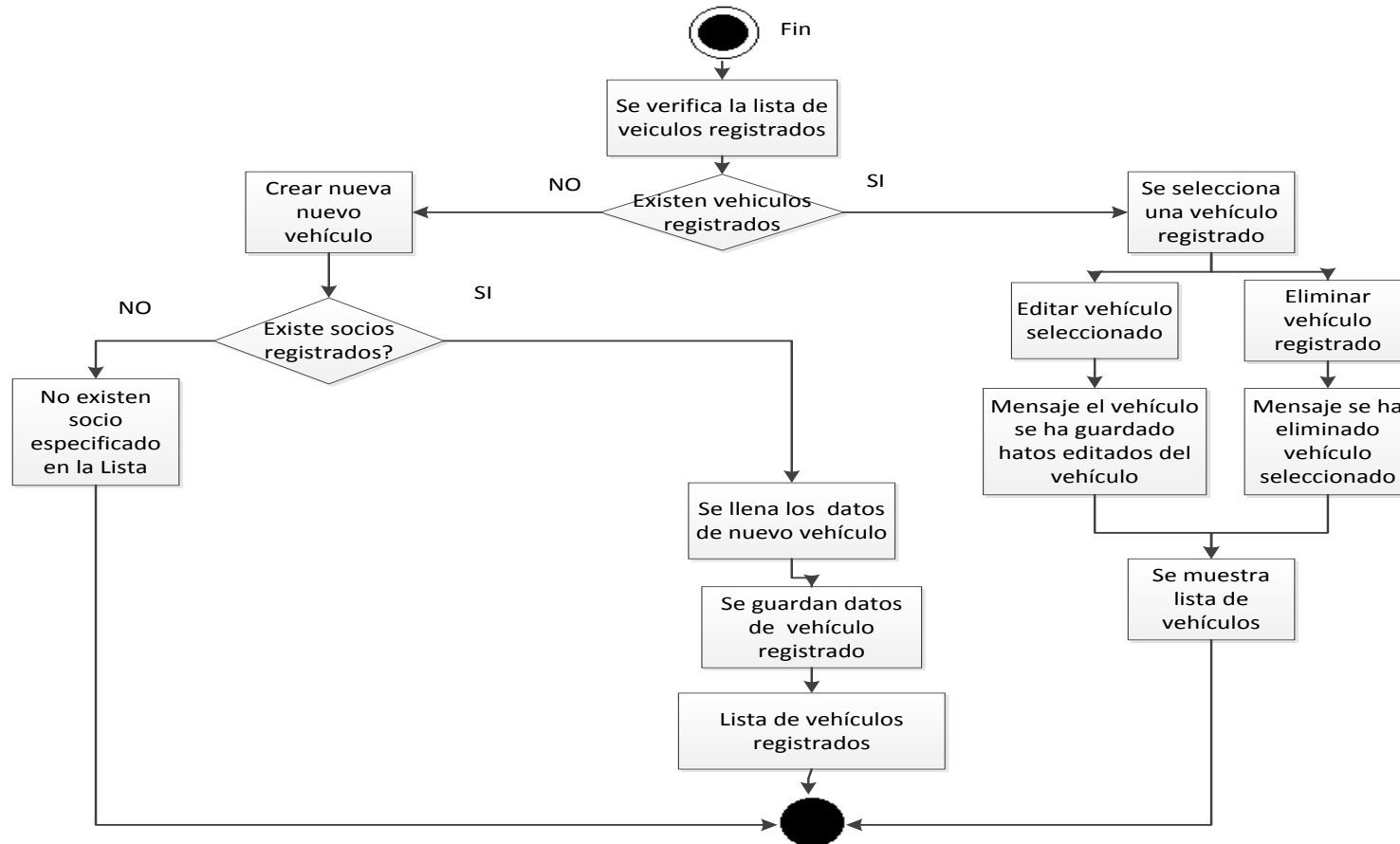
**Fuente.** Elaboración propia

**Figura 3.4**  
**Gestionar Chofer**



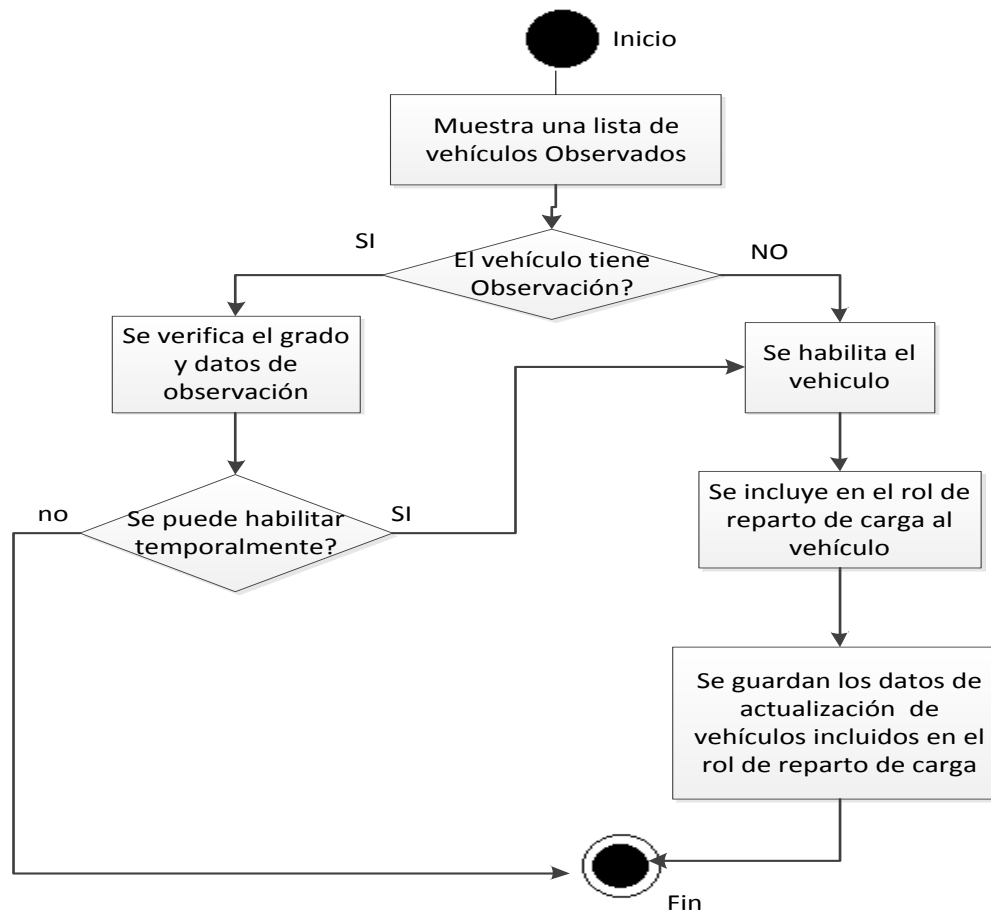
**Fuente.** Elaboración propia

**Figura 3.5**  
**Gestionar Vehículo**



**Fuente.** Elaboración propia

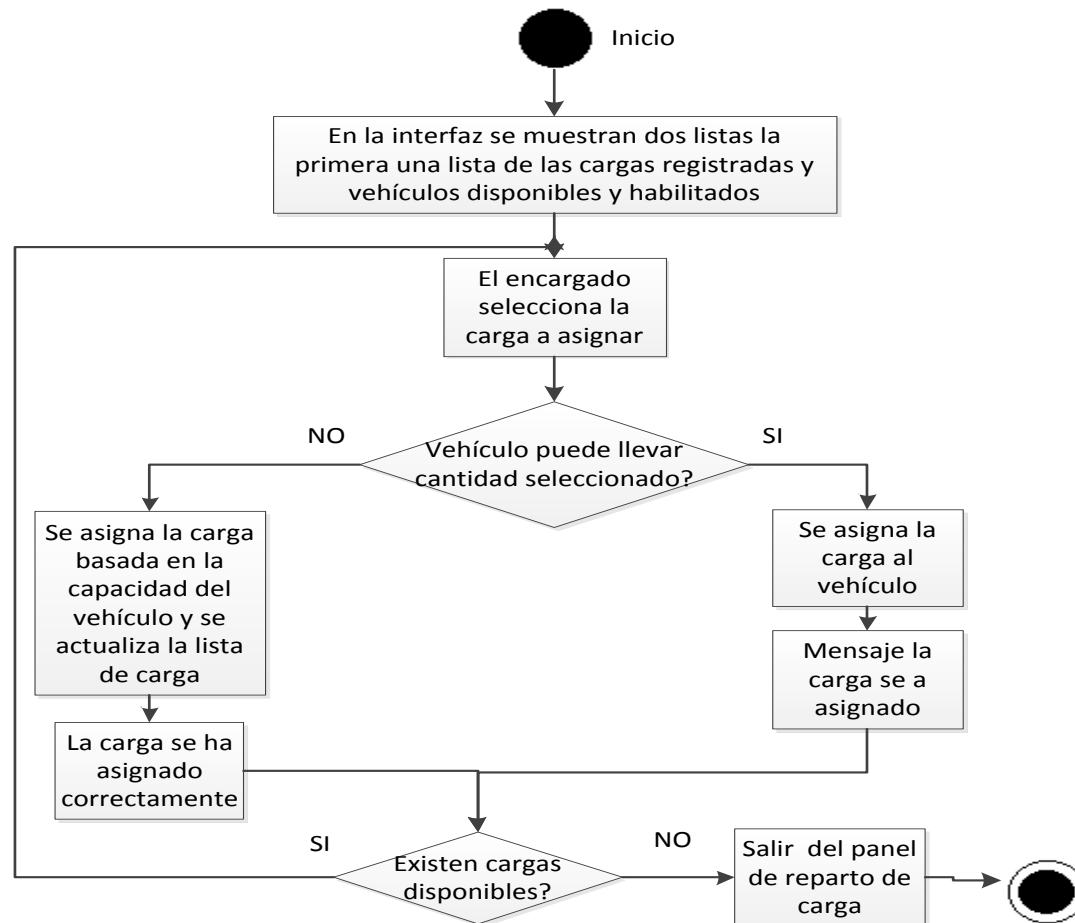
**Figura 3.6**  
**Habilitación de Vehículo**



**Fuente.** Elaboración propia



**Figura 3.7**  
**Asignación de carga**



**Fuente.** Elaboración propia

### 3.2 Planificación de iteración para el desarrollo de software

La presente fase se prioriza cada una de las historias de usuario, se estima el tiempo requerido como el esfuerzo para cada historia de usuario y para concluir se diseña el cronograma de entregas.

Para el desarrollo del siguiente proyecto se ha dividido en las siguientes Iteraciones.

**Tabla N° 3.1**  
**Especificación de módulos e Iteraciones**

N°	ITERACIÓN	N°	HISTORIAS DE USUARIOS
1	Gestión de socio, vehículo y choferes	1	Registro de socio
		2	Registro de chofer
		3	Registro de vehículo
		4	Registro de tipo de vehículo
		5	Registro de observaciones
		6	Gestionar cuentas de usuarios
2	Gestión de asignación y envío de carga	7	Gestión de carga
		8	Asignación de carga
3	Monitoreo vehicular	9	Configuración de receptor
		10	Control de monitoreo vehicular

**Fuente:** Elaboración propia

### 3.3 Primera iteración Desarrollar un módulo para la gestión de Vehículos, Socios y choferes

#### 3.3.1 Historias de Usuario.

Para esta fase se utilizarán las siguientes historias de usuarios como referencia principal para la identificación de requerimientos funcionales del sistema.

Las historias de usuarios en el desarrollo de proyecto XP, permiten especificar que requerimientos de usuario son necesarios para el desarrollo de la aplicación mediante la utilización de tarjetas en las cuales se describen las características que el sistema debe poseer. Las historias de usuario recopilados describen apenas que es lo que se quiere, esto están sujetos a modificaciones, e incluso a división:

##### 3.3.1.1 Historia de usuario registro de Socios.

**Tabla N 3.2**  
**Historia de usuario Gestión de Socios**

HISTORIA DE USUARIO	
<b>Número:</b> 1	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de Socio.	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registrar datos de un socio, visualizar la lista de socios registrados, modificar datos de los socios registrados y eliminar un socio registrado en el sistema.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.3.1.2 Historia de Usuario Gestión de choferes

**Tabla N 3.3**  
**Historia de usuario Gestión de Choferes**

HISTORIA DE USUARIO	
<b>Número:</b> 2	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de Choferes.	
<b>Prioridad en el negocio:</b> Media (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registrar datos de un chofer, visualizar la lista de choferes registrados, modificar datos de un choferes registrado y eliminar un socio registrado en el sistema.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.3.1.3 Historia de usuario Registro de Vehículo

**Tabla N 3.4**  
**Historia de usuario Gestión de Vehículos**

HISTORIA DE USUARIO	
<b>Número:</b> 3	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de Vehículos	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registrar datos de un vehículo, visualizar la lista de vehículos registrados, modificar datos de un vehículo registrado y eliminar un vehículo registrado en el sistema.	
<b>Observaciones:</b>	

**Fuente.** Elaboración propia

### 3.3.1.4 Historia de usuario Gestión de tipo de vehículos

**Tabla N 3.5**  
**Historia de usuario Gestión de Tipo Vehículos**

HISTORIA DE USUARIO	
<b>Número:</b> 4	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de Tipos Vehículo	
<b>Prioridad en el negocio:</b> Media (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registrar datos de un tipo de vehículo, visualizar la lista de tipos vehículos registrados, modificar datos de un tipo vehículo registrado y eliminar un vehículo registrado en el sistema.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.3.1.5 Historia de usuario Gestión Observaciones

**Tabla N 3.6**  
**Historia de usuario Gestión de Observaciones**

HISTORIA DE USUARIO	
<b>Número:</b> 5	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de observaciones	
<b>Prioridad en el negocio:</b> Media (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar la siguiente operación: Registrar las observaciones que se realizan a los socios y vehículos, Listar las observaciones, editar observaciones y eliminar observaciones.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.3.1.6 Historia de usuario Gestión de cuentas de usuario

**Tabla N 3.7**  
**Historia de usuario Gestión de cuentas de usuario**

HISTORIA DE USUARIO	
<b>Número:</b> 6	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de Requisitos	
<b>Prioridad en el negocio:</b> Media (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde el sistema permita registrar los requisitos presentados por los socios al momento de afiliarse al sindicato.	
<b>Observaciones:</b> Ninguna	

**Fuente:** Elaboración propia

### 3.3.1.7 Historia de usuario Gestión de Requisitos

**Tabla N 3.8**  
**Historia de usuario Gestión de cuentas de usuarios**

HISTORIA DE USUARIO	
<b>Número:</b> 7	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Gestión de cuentas de usuario	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Media (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde permita crear cuentas de cada socio el cual les permitirá ingresar al sistema, en caso de olvido tendría que tener las opciones de editar, o en algunos casos eliminar la cuenta.	
<b>Observaciones:</b> Ninguna	

**Fuente:** Elaboración propia

### **3.3.2 Planificación**

En la presente fase se prioriza cada una de las historias de usuario, se estima el tiempo requerido como el esfuerzo requerido para cada historia de usuario. Y para concluir se diseña el cronograma de entregas.

#### **3.3.2.1 Estimación de esfuerzo**

En esta fase se establece la prioridad para cada historia de usuario así como una estimación del esfuerzo necesario de cada una de ella con el fin de determinar un cronograma de entregas.

Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto.

Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, se mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Sin embargo, para la planificación del tiempo requerido para las entregas de las iteraciones que se utilizaran en desarrollo del sistema se utilizara la siguiente tabla de estimación de tiempo.



**Tabla 3.9**

Modelo para estimación de tiempo

HORAS	DÍAS	SEMANAS
8 Horas	5 días	4 semanas
Son las horas que se van a dedicar al desarrollo del proyecto	Son los días que se va a dedicar al desarrollo del proyecto	Semanas al mes que se va a dedicar al desarrollo proyecto

**Fuente.** Elaboración Propia

Determinado el tiempo de estimación, se procede a la elaboración del plan de entrega, mediante la utilización de la estimación por historias que se utilizara en cada iteración.

En la siguiente tabla se muestra la estimación de tiempo de esfuerzo que se necesitan para el desarrollo de la respectiva iteración.

**Tabla 3.10****Estimación de esfuerzo de la primera iteración.**

Nro.	ITERACIÓN	Nº	HISTORIAS DE USUARIOS	TIEMPO ESTIMADO		
				Sem.	Día	Hrs
1	Gestión de socio, vehículo y choferes	1	Registro de socio	1	4	32
		2	Registro de chofer	1	4	32
		3	Registro de vehículo	1	3	24
		4	Registro de tipo de vehículo	1	3	24
		5	Registro de observaciones	2	6	48
		6	Gestionar cuentas de usuarios	1	3	24
	<b>Total</b>			<b>7</b>	<b>27</b>	<b>184</b>

**Fuente.** Elaboración propia

### 3.3.3 Diagrama de caso de uso General Primera Iteración

**Figura: 3.8**

**Diagramas de caso de uso primera iteración**

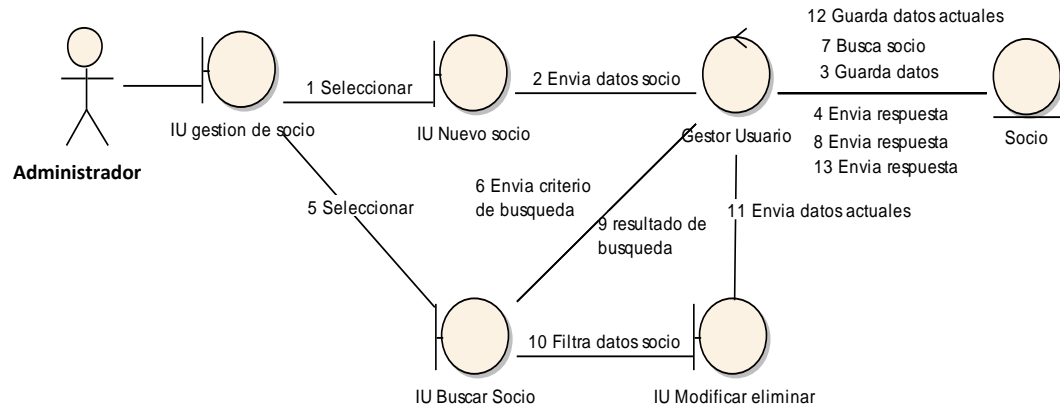


**Fuente:** Elaboración Propia

### 3.3.3.1 Diagramas de colaboración

**Figura 3.9**

#### Diagrama de colaboración Gestión de socio

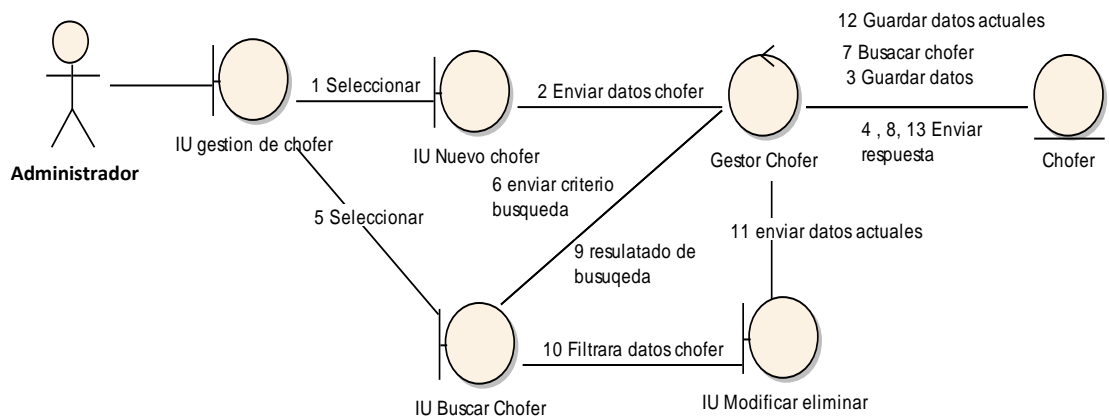


**Fuente.** Elaboración propia

### 3.3.3.2 Diagrama de colaboración gestionar chofer

**Figura 3.10**

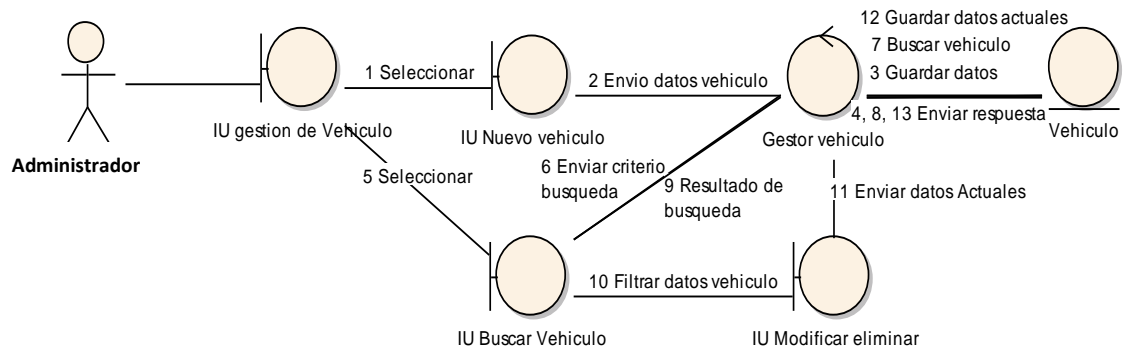
#### Diagrama de colaboración Gestión de chofer



**Fuente.** Elaboración propia

### 3.3.3.3 Diagrama de colaboración gestión de vehículos

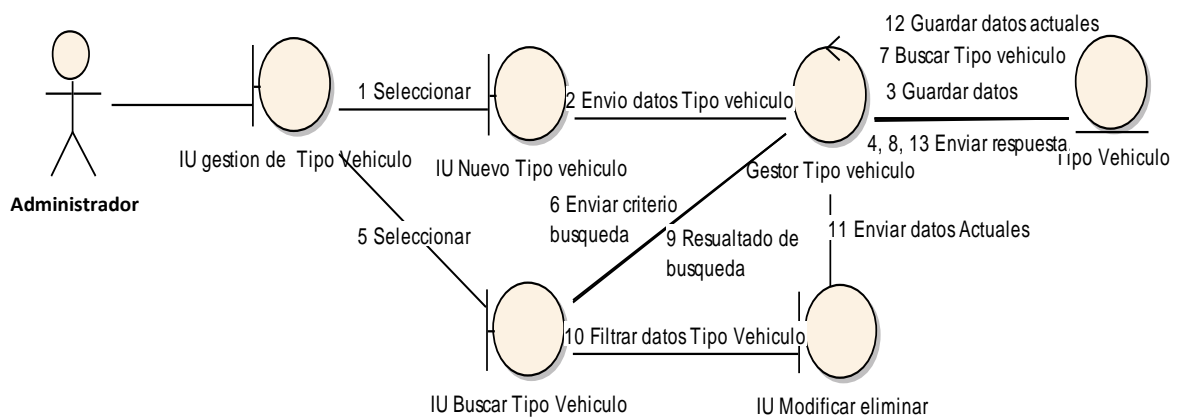
**Figura 3.11**  
**Diagrama de colaboración Gestión de vehículos**



**Fuente.** Elaboración propia

### 3.3.3.4 Diagrama de colaboración gestión de Tipo vehículo

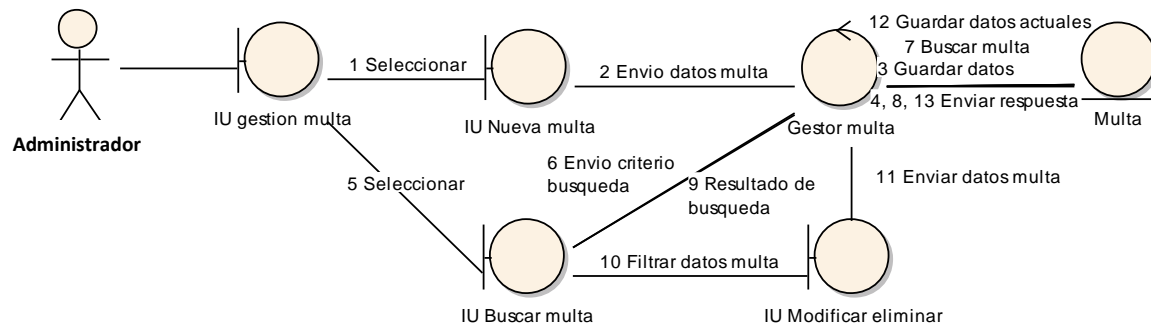
**Figura 3.12**  
**Diagrama de colaboración Gestión de tipo vehículos**



**Fuente.** Elaboración propia

### 3.3.3.5 Diagrama de colaboración gestión de Observaciones

**Figura 3.13**  
**Diagrama de colaboración gestión de observaciones**



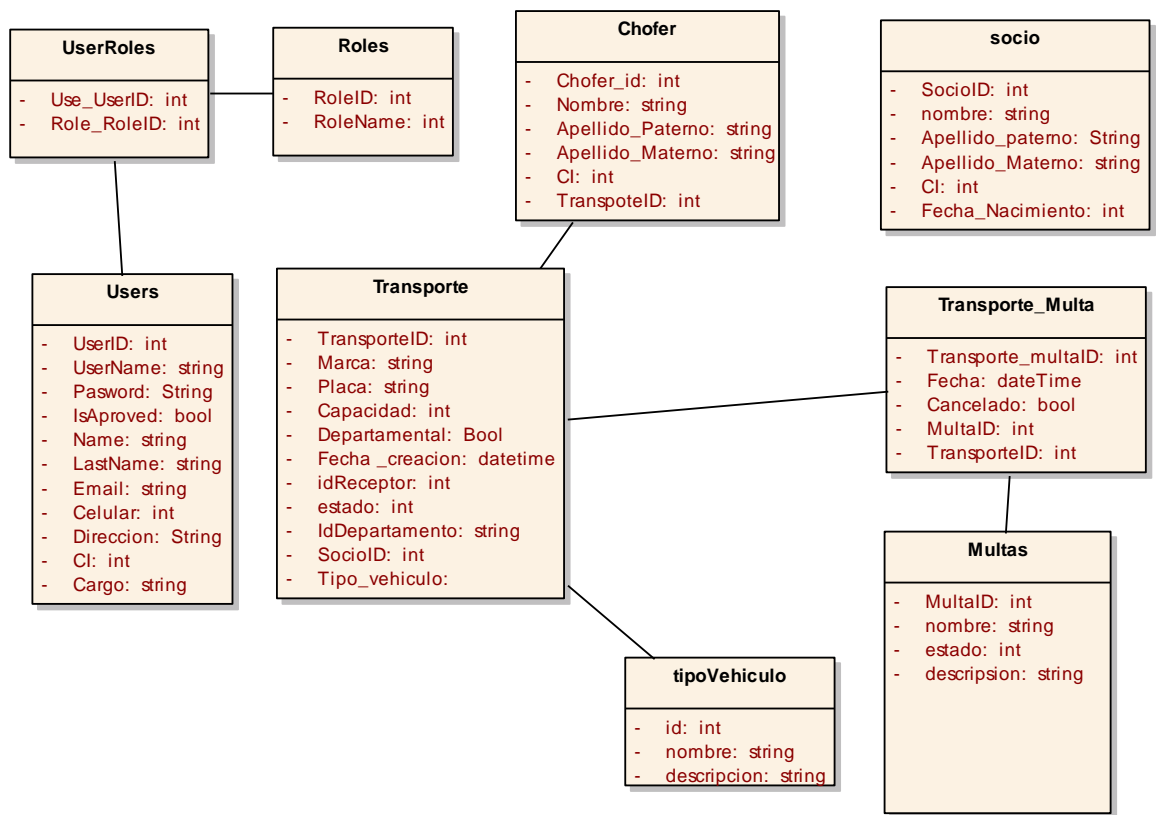
**Fuente.** Elaboración propia

### 3.3.4 Diseño

Para el diseño se vio conveniente realizar un modelo de Entidad-Relación la cual ayudará para la creación de la Base de datos, de las clases y las vistas que se necesitarán.

A continuación en la siguiente figura se mostrará el modelo entidad - relación para la gestión de socios, choferes y vehículos.

**Figura 3.14**  
**Modelo Entidad de relación**



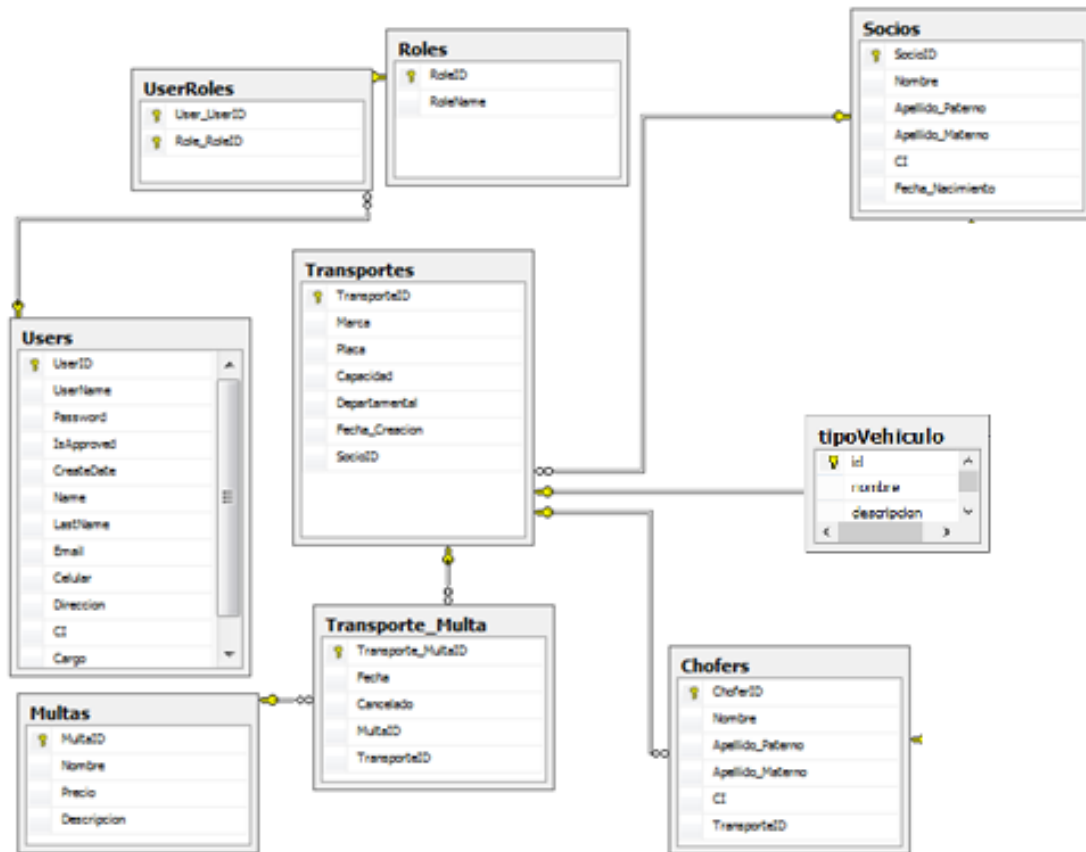
**Fuente.** Elaboración propia.

#### 3.3.4.1 Diseño físico de la Base de datos de la primera iteración

El diseño físico parte del esquema global obteniendo del modelo lógico, en él se obtiene la descripción de la implementación de la base de datos

A continuación en la siguiente figura se muestra el diseño físico del sistema que fue implementado en la base de datos.

**Figura 3.15**  
**Diseño físico del sistema para la primera iteración**



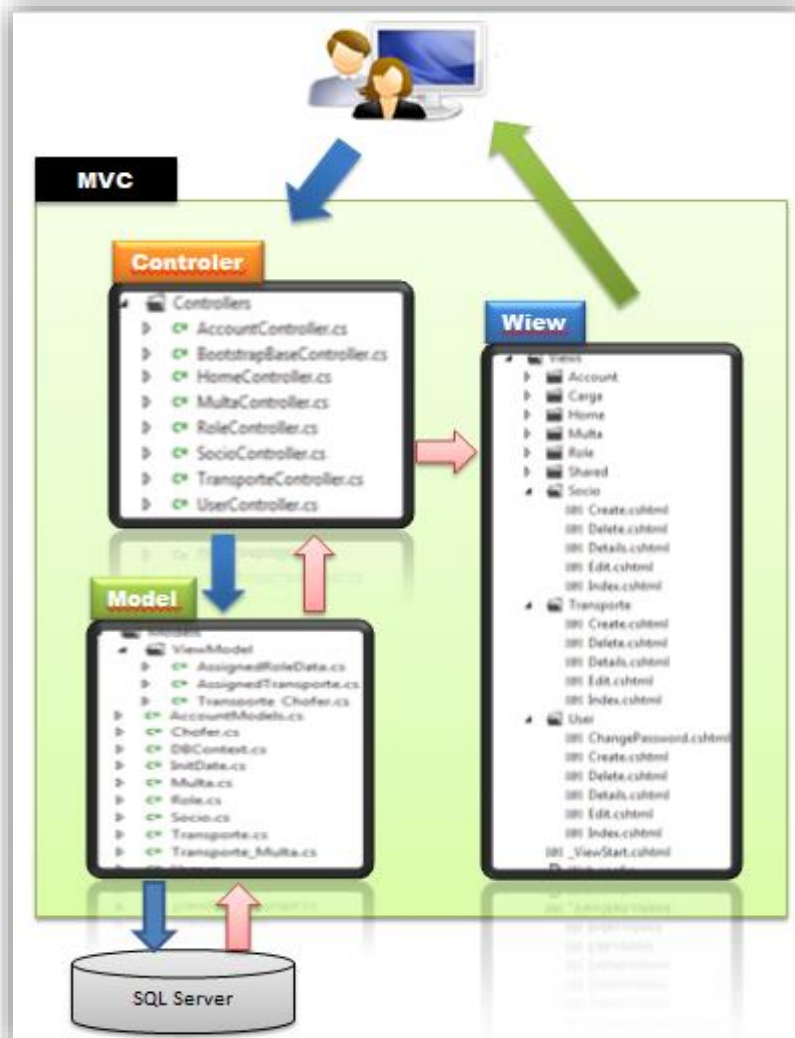
**Fuente.** Elaboración propia

### 3.3.5 Arquitectura del sistema

A continuación se muestra la implementación de la arquitectura MVC4 en la primera Iteración del proyecto.

**Figura 3.16**

**Implementación de la Arquitectura del sistema en la primera Iteración**



**Fuente** Elaboración propia.

### 3.3.6 Codificación

En esta fase se iniciará con el desarrollo del sistema de la primera iteración aplicando las anteriores fases de análisis y diseño, se aportará el lenguaje de programación y la base de datos.

Para el proceso de codificación del CRUD Para el subsistema de gestionar usuarios, gestionar Socio, gestionar vehículos, gestionar choferes y gestionar multas.



### 3.3.6.1 Modelo

En el modelo se define los atributo (datos) lo cual vemos en la siguiente figura.

**Figura 3.17**  
**Modelo administrar Vehículo**

```
namespace DisCar.Models
{
    public class Transporte
    {
        [Key]
        public int TransporteID { get; set; }

        [Required]
        public string Marca { get; set; }

        [Required]
        public string Placa { get; set; }

        [Required]
        public int Capacidad { get; set; }

        public bool Departamental { get; set; }

        public DateTime? Fecha_Creacion { get; set; }

        public int? SocioID { get; set; }

        public virtual ICollection<Transporte_Multa> Transporte_Multas { get; set; }
        public virtual ICollection<Distribucion> Distribuciones { get; set; }
        public virtual ICollection<Chofer> Choferes { get; set; }
        public virtual Socio Socio { get; set; }
    }
}
```

**Fuente.** Sistema desarrollado

### 3.3.6.2 Controlador

El proceso que sigue cuando se realiza una petición de algún método es la consulta al controlador, a continuación se muestra el código de creación de un nuevo vehículo.

**Figura 3.18**  
**Método Crear Vehículo**

```
// GET: /Transporte/Create
public ActionResult Create(int id)
{
    ViewBag.id = id;

    return View();
}

//
// POST: /Transporte/Create
[HttpPost]
public ActionResult Create(Transporte_Chofer tc, int id = 0)
{
    if (ModelState.IsValid && id != 0)
    {
        Transporte traE = db.Transportes.Where(c => c.Placa.ToUpper().Equals(tc.Transporte.Placa.ToUpper())).FirstOrDefault();

        if (traE == null)
        {
            tc.Transporte.Placa = tc.Transporte.Placa.ToUpper();
            tc.Transporte.Fecha_Creacion = DateTime.Now;
            tc.Transporte.SocioID = id;
            db.Transportes.Add(tc.Transporte);

            db.SaveChanges();

            tc.Chofer.TransporteID = db.Transportes.OrderByDescending(c => c.TransporteID).FirstOrDefault().TransporteID;

            db.Chofers.Add(tc.Chofer);
            db.SaveChanges();
        }
        return RedirectToAction("Index", new { id = id });
    }
}
```

**Fuente.** Sistema desarrollado

En la siguiente figura se detalla la operación editar de algún vehículo, obteniendo su id para la verificación de la existencia.

**Figura 3.19**  
**Método editar Vehículo**

```
public ActionResult Edit(int id = 0)
{
    Transporte transporte = db.Transportes.Find(id);
    if (transporte == null)
    {
        return HttpNotFound();
    }
    ViewBag.SocioID = new SelectList(db.Socios, "SocioID", "Nombre", transporte.SocioID);
    return View(transporte);
}

//
// POST: /Transporte/Edit/5
[HttpPost]
public ActionResult Edit(Transporte transporte)
{
    if (ModelState.IsValid)
    {
        db.Entry(transporte).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.SocioID = new SelectList(db.Socios, "SocioID", "Nombre", transporte.SocioID);
    return View(transporte);
}
```

**Fuente.** Sistema desarrollado

En la siguiente figura detalla el código de eliminar un vehículo obteniendo su id para la verificación de su existencia.

**Figura 3.20**  
**Método eliminar Vehículo**

```
// GET: /Transporte/Delete/5

public ActionResult Delete(int id = 0)
{
    Transporte transporte = db.Transportes.Find(id);
    if (transporte == null)
    {
        return HttpNotFound();
    }
    return View(transporte);
}

//
// POST: /Transporte/Delete/5

[HttpPost, ActionName("Delete")]
public ActionResult DeleteConfirmed(int id)
{
    Transporte transporte = db.Transportes.Find(id);
    db.Transportes.Remove(transporte);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```


**Fuente.** Sistema desarrollado

#### 3.3.6.3 Vista

Es la interfaz donde se puede comunicar al usuario con el sistema, para esta parte se tomará como ejemplo la creación de un vehículo el mismo que tiene su propio controlador y modelo.

**Figura 3.21**  
**Vista de creación de Vehículo**

ASOTRAM 1.0
Inicio




### Datos del transporte

Marca

Placa

Capacidad

Departamental  
☐



### Datos Personales

Nombre

Apellido\_Paterno

Apellido\_Materno

CI

**Fuente.** Sistema desarrollado

**Figura 3.22**  
**Vista lista de vehículos**

ASOTRAM 1.0
Inicio
Usuarios
Transportes ▾
Asignación de Carga ▾
Reportes ▾

### Lista de transportes en espera

Orden	Estado	Chofer	Placa	Capacidad	Departamental
1	Disponible	Rivaldo Rojas Riola	KJI-458	500	<input checked="" type="checkbox"/>
2	Disponible	Juan Granado Villa	LG-895	500	<input checked="" type="checkbox"/>
3	Disponible	Martin Charcas Rioja	LK-8001	250	<input type="checkbox"/>
4	Disponible	Ronald quispe Ricaldis	LP-895	250	<input type="checkbox"/>

© ASOTRAM 2014


**Fuente.** Sistema desarrollado

**Figura 3.23**  
**Vista de creación de Socio**

**ASOTRAM 1.0** Inicio

## Creando socio

### Datos del socio



Nombre

Apellido\_Paterno

Apellido\_Materno

CI

Fecha\_Nacimiento

**Crear**

[Regresar a la lista](#)

**Fuente.** Sistema desarrollado

**Figura 3.24**  
**Vista lista de socios**

ASOTRAM 1.0 Inicio [Iniciar sesión](#)

### Lista de socios

Crear Nuevo Socio

Nombre	Apellido_Paterno	Apellido_Materno	CI	Fecha_Nacimiento	
Pablo	Huanca	Morales	4856965	02/05/1978 12:00:00 a.m.	<a href="#">Editar</a>   <a href="#">Trasportes</a>   <a href="#">Eliminar</a>
Rita	Alcocer	Malicia	4896568	02/05/1960 12:00:00 a.m.	<a href="#">Editar</a>   <a href="#">Trasportes</a>   <a href="#">Eliminar</a>
Lourdes	Rivales	Canino	6956899	02/08/1970 12:00:00 a.m.	<a href="#">Editar</a>   <a href="#">Trasportes</a>   <a href="#">Eliminar</a>
Ricardo	Poma	Lopez	6958788	02/05/1988 12:00:00 a.m.	<a href="#">Editar</a>   <a href="#">Trasportes</a>   <a href="#">Eliminar</a>

© ASOTRAM 2014

**Fuente.** Sistema desarrollado

### 3.3.7 Pruebas

En esta fase de prueba del sistema, se realizara pruebas unitarias aplicadas en el desarrollo concluido de la primera iteración.

#### 3.3.7.1 Pruebas unitarias

Es sistema es construido a base de reglas establecidas en la fase de análisis donde cada regla se convierte en prueba (test), para garantizar el perfecto funcionamiento del método y del sistema. A continuación se describirá alguno de los métodos

**Figura 3.25**  
**Pruebas de la primera iteración**

```
[TestMethod]
public void TransportesMultados()
{
    Transporte_Chofer tc = new Transporte_Chofer();
    int id = 0;
    Transporte traE = db.Transportes.Where(c => c.Placa.ToUpper().Equals(tc.Transporte.Placa.ToUpper())).FirstOrDefault();

    if (traE == null)
    {
        tc.Transporte.Placa = tc.Transporte.Placa.ToUpper();
        tc.Transporte.Fecha_Creacion = DateTime.Now;
        tc.Transporte.SocioID = id;
        db.Transportes.Add(tc.Transporte);

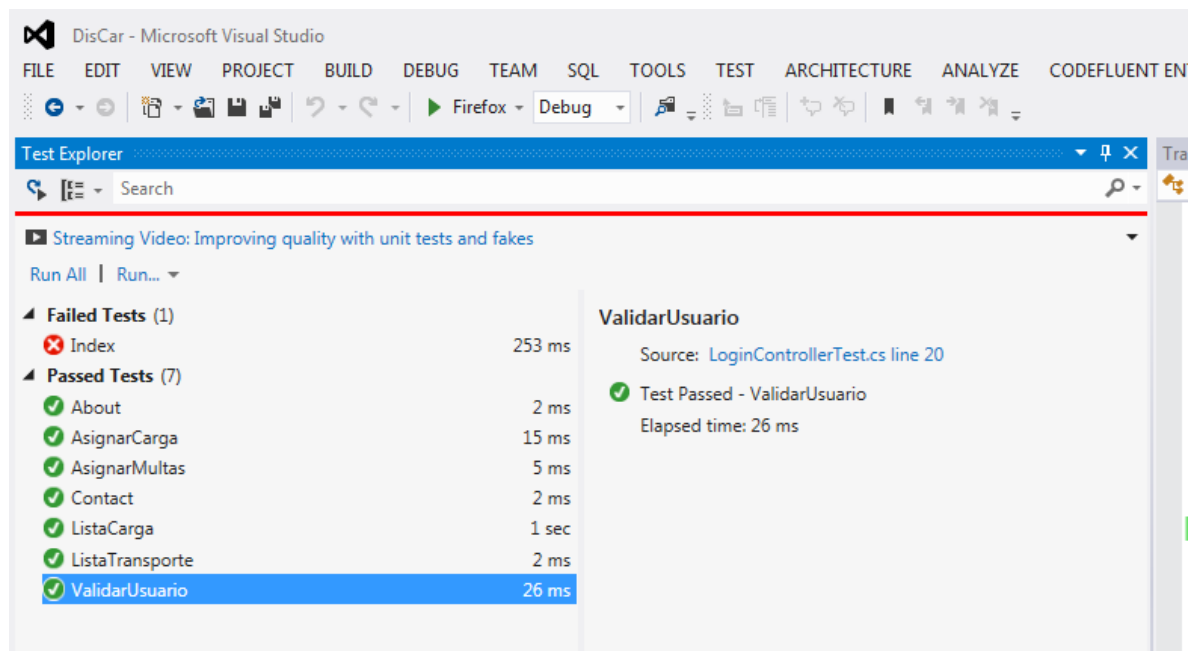
        db.SaveChanges();

        tc.Chofer.TransporteID = db.Transportes.OrderByDescending(c => c.TransporteID).FirstOrDefault().TransporteID;

        db.Chofers.Add(tc.Chofer);
        db.SaveChanges();
    }
    Assert.IsNotNull(traE);
}
```

**Fuente.** Sistema desarrollado

**Figura 3.26**  
**Resultado de prueba de la primera iteración**



**Fuente.** Sistema desarrollado

### 3.4 Segunda Iteración Gestión de asignación y envío de carga

#### 3.4.1 Historia de usuario

##### 3.4.1.1 Historia de usuario Gestión de carga

**Tabla N 3.11**

**Historia de usuario Registro de carga**

HISTORIA DE USUARIO	
<b>Número:</b> 8	<b>Usuario:</b> Presidente de la Asociación y secretaria
<b>Nombre de la historia:</b> Gestión de carga	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Alta (Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registro de una carga con sus respectivos datos, en caso de que tenga un error en los datos me permita realizar las operaciones de editar, eliminar y por ultimo pueda visualizar la lista cargas registradas.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

##### 3.4.1.2 Historia de usuario Asignación de carga

**Tabla N 3.12**

**Historia de usuario asignación de carga**

HISTORIA DE USUARIO	
<b>Numero:</b> 1	<b>Usuario:</b> Presidente de la Asociación
<b>Nombre de la historia:</b> Asignación de carga.	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Alta (Alta/media/Baja)
<b>Descripción:</b> Se requiere un módulo para la asignación de cargas registradas en el sistema el cual pueda permitir realizar la asignación de manera eficiente, en base a las	



capacidades de tonelaje de transporte de carga de los vehículos.
<b>Observaciones:</b> Ninguna

**Fuente.** Elaboración propia

### 3.4.2 Planificación de entrega

**Tabla 3.13**

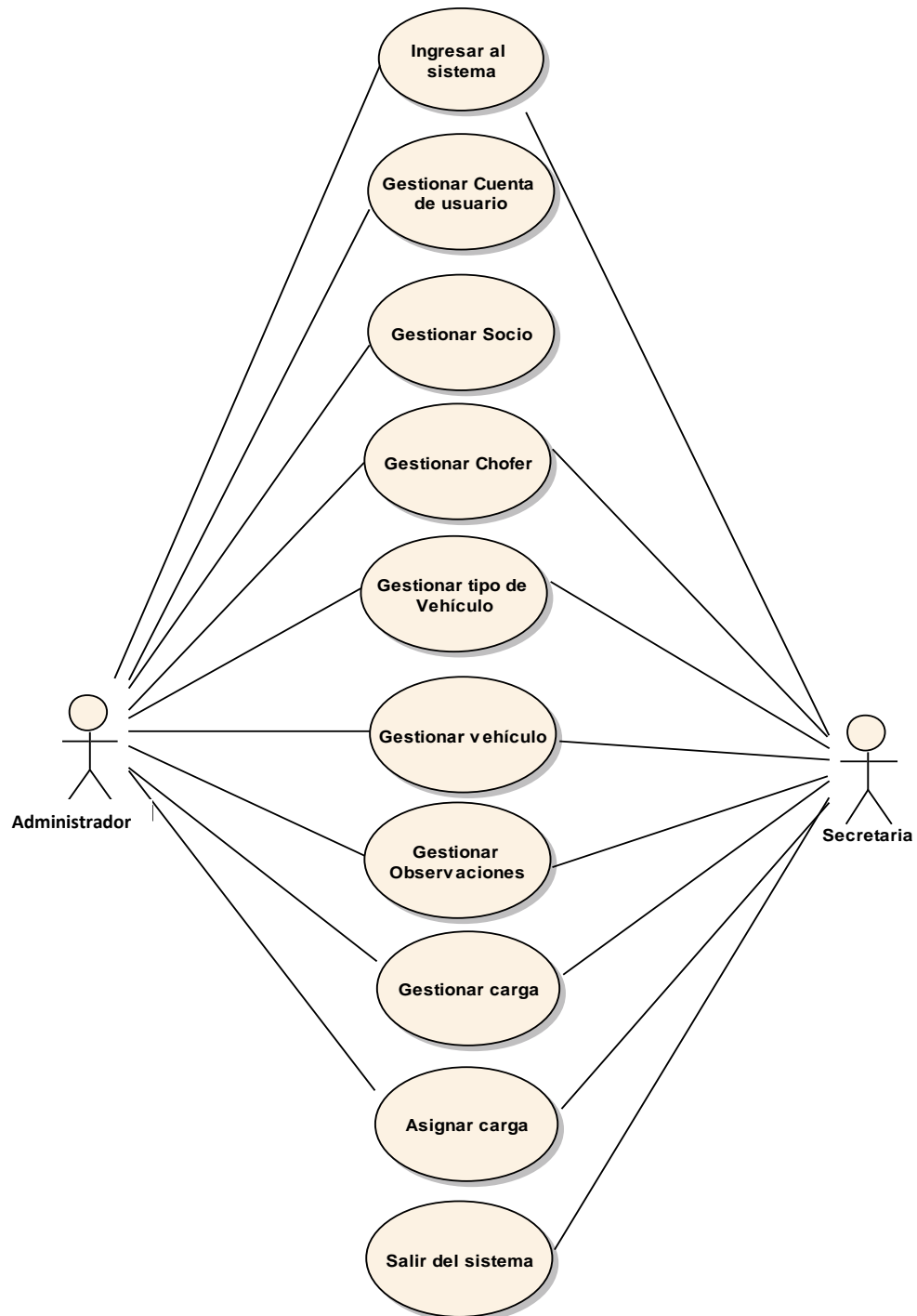
**Estimación de esfuerzo de la segunda iteración.**

Nº	ITERACIÓN	Nº	HISTORIAS DE USUARIOS	TIEMPO ESTIMADO		
				Sem.	Día	Hrs
2	Gestión y envió de carga	8	Gestión de cargas	1	5	40
		9	Asignación de carga	1	5	40
Tiempo total estimado				3	10	80

**Fuente.** Elaboración propia

### 3.4.3 Diagrama de caso de uso General segunda Iteración

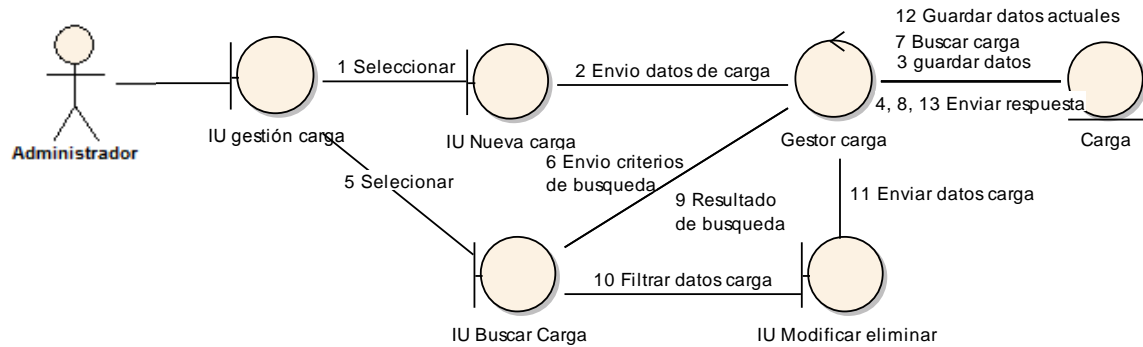
**Figura: 3.27**  
**Diagramas de caso de uso segunda iteración**



**Fuente.** Elaboración Propia

### 3.4.3.1 Diagramas de colaboración gestionar carga

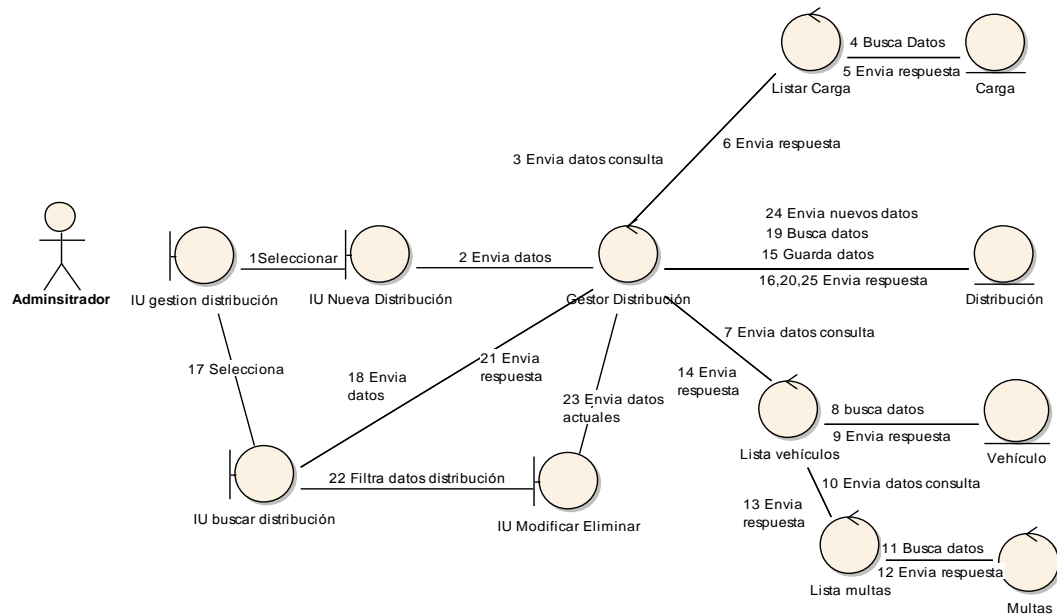
**Figura 3.28**  
**Diagrama de colaboración Gestión de carga**



**Fuente.** Elaboración propia

### 3.4.3.2 Diagrama de colaboración Asignación de carga

**Figura 3.29**  
**Diagrama de colaboración Asignación de carga**



**Fuente.** Elaboración propia

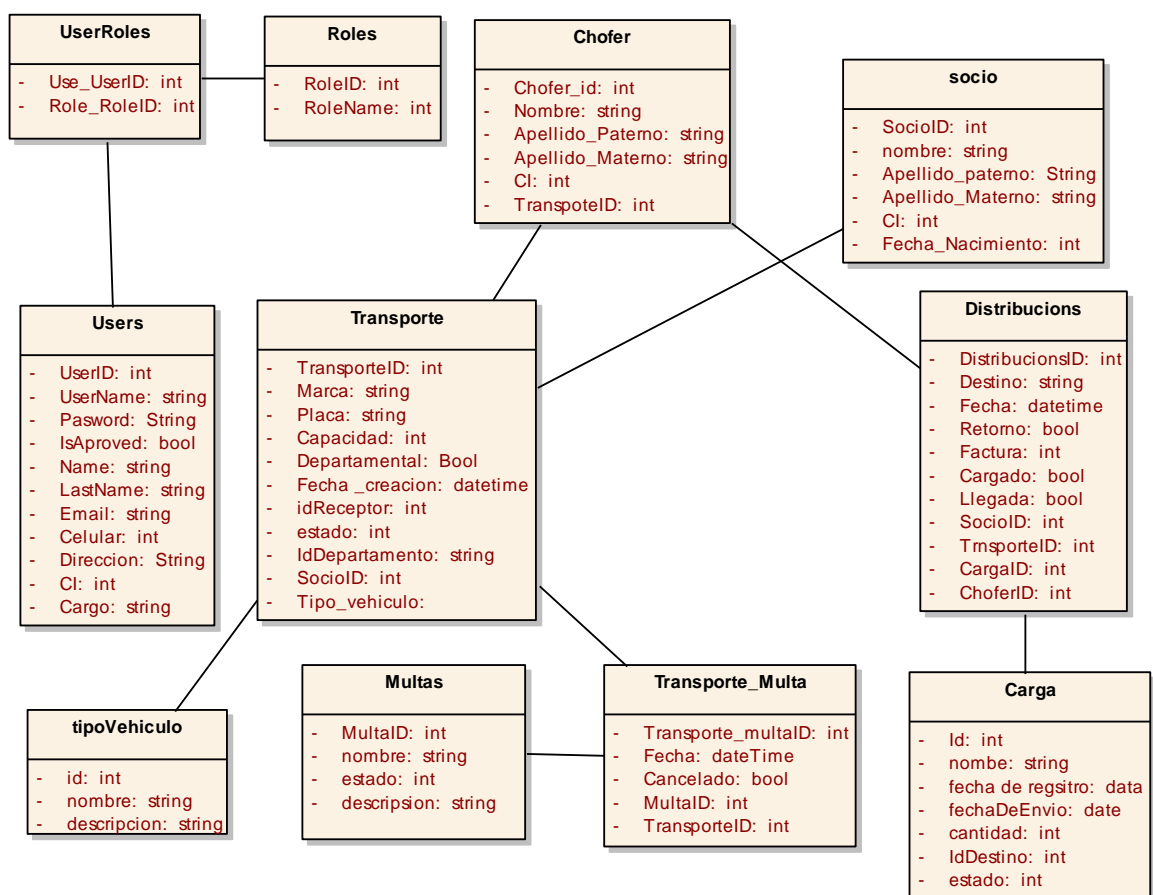
### 3.4.4 Diseño

Para el diseño se vio conveniente realizar un modelo de Entidad-Relación la cual ayudara para la creación de la Base de datos, de las clases y las vistas que se necesitaran.

A continuación en la siguiente figura se mostrará el modelo entidad - relación para la gestión de cargas y reparto de las mismas.

**Figura 3.30**

**Modelo Entidad de relación segunda iteración**



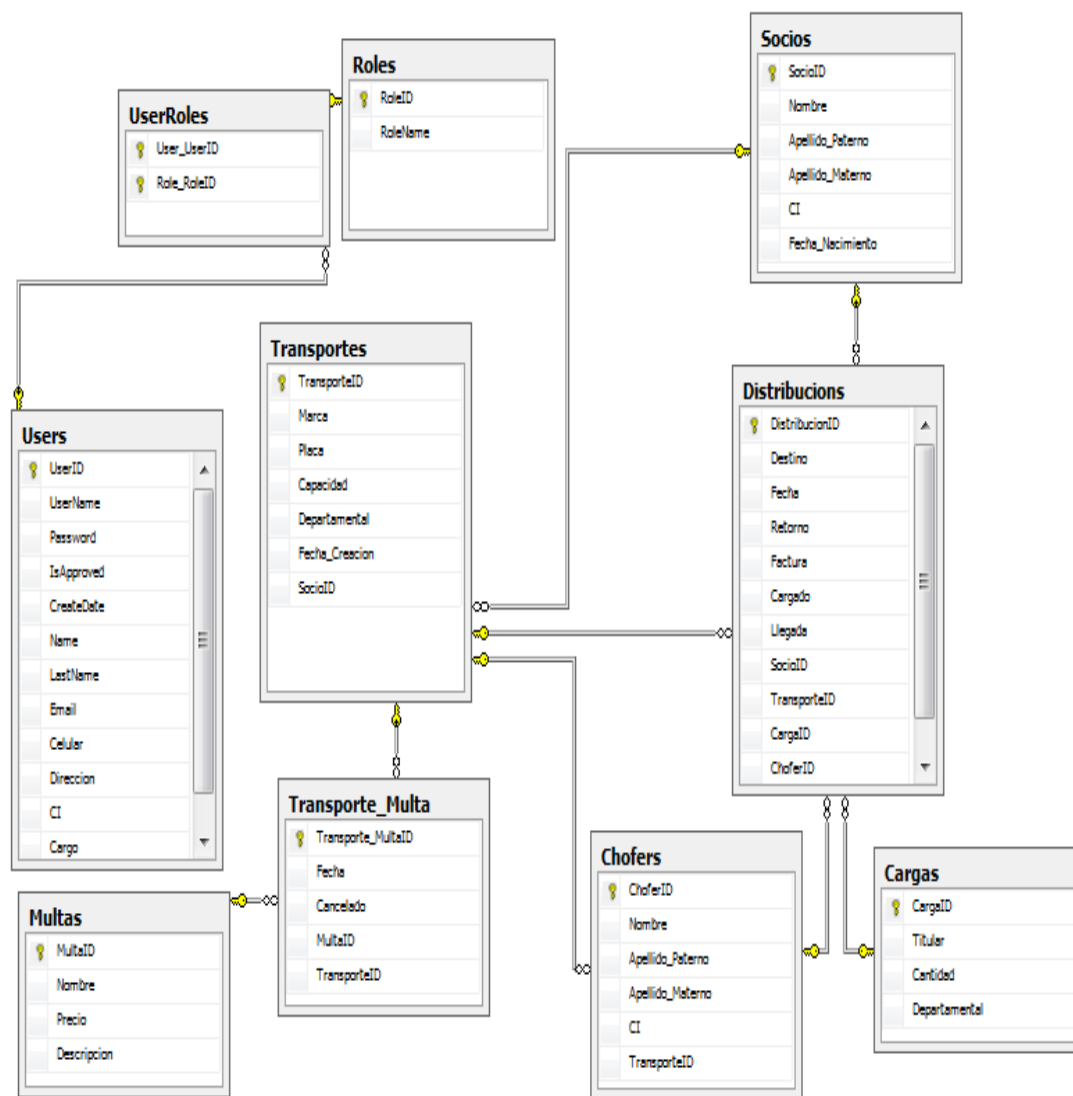
**Fuente.** Elaboración propia.

#### 3.4.4.1 Diseño físico de la Base de datos de la Segunda iteración

El diseño físico parte del esquema global obteniendo del modelo lógico, en él se obtiene la descripción de la implementación de la base de datos

A continuación en la siguiente figura se muestra el diseño físico del sistema que fue implementado en la base de datos.

**Figura 3.31**  
**Diseño físico del sistema para la segunda iteración**



**Fuente.** Elaboración propia

### 3.4.5 Codificación

En la parte de la codificación se seguirá con la arquitectura seleccionada, en el proceso de gestión y asignación de carga.

#### 3.4.5.1 Modelo

En el modelo se define los atributo (datos) lo cual vemos en la siguiente figura.

**Figura 3.32**  
**Modelo asignación de carga**

```
namespace DisCar.Models
{
    public class Distribucion
    {
        [Key]
        public int DistribucionID { get; set; }
        public string Destino { get; set; }
        public DateTime Fecha { get; set; }
        public bool? Retorno { get; set; }
        public string Factura { get; set; }
        public string Cargado { get; set; }
        public DateTime? Llegada { get; set; }
        public int SocioID { get; set; }
        public int TransporteID { get; set; }
        public int CargaID { get; set; }
        public int ChoferID { get; set; }

        public virtual Socio Socio { get; set; }
        public virtual Transporte Transporte { get; set; }
        public virtual Carga Carga { get; set; }
        public virtual Chofer Chofer { get; set; }
    }
}
```

**Fuente.** Sistema desarrollado

#### 3.4.5.2 Controlador

El proceso que sigue cuando se realiza una petición de algún método es la consulta al controlador, a continuación se muestra el código de creación de un nuevo vehículo.

**Figura 3.33**  
**Controlador método asignar carga**

```
public ActionResult IndexAsignarCarga(int id)
{
    List<Distribucion> distribuciones = db.Distribuciones.Include(c => c.Chofer).Include(c => c.Transporte).Where(x => x.CargaID == id).ToList();

    if (distribuciones.Count == 0 )
    {
        List<AssignedTransporte> habilitados = new List<AssignedTransporte>();
        List<AssignedTransporte> observados = new List<AssignedTransporte>();
        ViewBag.id = id;

        Carga carga = db.Cargas.Where(c => c.CargaID == id).FirstOrDefault();

        foreach (var item in db.Transportes.Include(c => c.Chofers).Include(c => c.Transporte_Multas).Include(c => c.Distribuciones))
        {
            AssignedTransporte asigne = new AssignedTransporte();
            asigne.Carga = carga;
            asigne.Asignacion = false;
            asigne.Capacidad = item.Capacidad;
            asigne.Chofer = item.Chofers.FirstOrDefault().Nombre + " " + item.Chofers.FirstOrDefault().Apellido_Paterno + " " + item.Chofers.FirstOrDefault().Apellido_Materno;
            asigne.Departamental = item.Departamental;
            asigne.Placa = item.Placa;
            asigne.TransporteID = item.TransporteID;

            bool ret = true;

            if (item.Distribuciones.Count > 0)
            {
                foreach (var item3 in item.Distribuciones.OrderByDescending(c => c.Llegada))
                {
                    asigne.Llegada = item3.Llegada;
                    if (item3.Retorno == false)
                    {
                        ret = false;
                    }
                }
            }
        }
    }
}
```

```

        else
        {
            asigne.Llegada = item.Fecha_Creacion;
        }

        asigne.Retorno = ret;

        bool flag = true;

        foreach (var item2 in item.Transporte_Multas)
        {
            if (!item2.Cancelado)
            {
                flag = false;
                break;
            }
        }

        if (flag && ret)
        {
            habilitados.Add(aspigne);
        }
        else
        {
            observados.Add(aspigne);
        }
    }

    int cargaTotal = carga.Cantidad;

    foreach (var item in habilitados.OrderBy(c => c.Llegada))
    {
        if (!carga.Departamental)
        {
            if (item.Capacidad <= cargaTotal && cargaTotal > 0)
            {
                cargaTotal -= item.Capacidad;
                item.Asignacion = true;
            }
        }
        else
        {
            if (item.Capacidad <= cargaTotal && cargaTotal > 0 && item.Departamental)
            {
                cargaTotal -= item.Capacidad;
                item.Asignacion = true;
            }
        }
    }

    ViewBag.Observados = observados;

    return View(habilitados.OrderBy(c => c.Llegada));
}
else
{
    ViewDistribucion distri = new ViewDistribucion();
    distri.Carga = db.Cargas.Where(x => x.CargaID == id).FirstOrDefault();
    distri.Distribuciones = distribuciones;

    return View("Details", distri);
}
}
}

```

**Fuente.** Sistema desarrollado



### 3.4.5.3 Vista

Es la interfaz donde se puede comunicar al usuario con el sistema, para esta parte se tomara como ejemplo la interfaz de asignación de carga a los vehículos.

**Figura 3.34**  
**Vista de Lista de cargas**

ASOTRAM 1.0 Inicio Usuarios Transportes ▾ Asignación de Carga ▾ Reportes ▾ <span>ronald</span> Cerrar sesión				
<b>Lista de cargas</b> <a href="#">Crear nueva Carga</a>				
Estado	Titular	Cantidad	Departamental	
Sin Distribución	COBOCE	1500	<input type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Entregado	coboce	500	<input checked="" type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Entregado	COBOCE	500	<input checked="" type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Sin Distribución	Coboce	700	<input type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Sin Distribución	coboce	1000	<input checked="" type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Entregado	COBOCE	500	<input type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>
Entregado	COBOCE	1000	<input checked="" type="checkbox"/>	<a href="#">Eliminar</a>   <a href="#">Distribucion</a>

**Fuente.** Sistema desarrollado

**Figura 3.35**  
**Vista Asignación de carga**

ASOTRAM 1.0 Inicio Usuarios Transportes ▾ Asignación de Carga ▾ Reportes ▾ <span>ronald</span> Cerrar sesión					
<b>Distribución de Carga</b> <b>COBOCE : 1500</b>					
Destino:	santa cruz	Nº de Factura:	545556985		
<b>Disponibles</b>					
Asignación	Chofer	Placa	Capacidad	Departamental	Última llegada
<input checked="" type="checkbox"/>	Rivaldo Rojas Riola	KJI-458	500	<input checked="" type="checkbox"/>	09/09/2014 04:10:14 p.m.
<input checked="" type="checkbox"/>	Juan Granado Villa	LG-895	500	<input checked="" type="checkbox"/>	09/09/2014 04:10:14 p.m.
<input checked="" type="checkbox"/>	Martin Charcas Rioja	LK-8001	250	<input type="checkbox"/>	09/09/2014 04:10:26 p.m.
<input checked="" type="checkbox"/>	Ronald quispe Ricaldis	LP-895	250	<input type="checkbox"/>	09/09/2014 04:10:26 p.m.
<a href="#">Asignar</a>					

**Fuente.** Sistema desarrollado

### 3.4.6 Pruebas

En esta fase de prueba del sistema, se realizará pruebas unitarias aplicadas en los desarrollos concluidos de la segunda iteración.

#### 3.4.6.1 Pruebas unitarias

Es sistema es construido a base de reglas establecidas en la fase de análisis donde cada regla se convierte en prueba (test), para garantizar el perfecto funcionamiento del método y del sistema.

**Figura 3.36**  
**Test método asignar carga**

```
[TestMethod]
public void AsignarCarga()
{
    var id = 1;
    Carga carga = db.Cargas.Where(c => c.CargaID == id).FirstOrDefault();

    foreach (var item in db.Transportes)
    {
        AssignedTransporte asigne = new AssignedTransporte();
        asigne.Carga = carga;
        asigne.Asignacion = false;
        asigne.Capacidad = item.Capacidad;
        asigne.Chofer = item.Chofers.FirstOrDefault().Nombre + " " + item.Chofers.FirstOrDefault().Apellido_Paterno + " " + item.Chofers.FirstOrDefault().Apellido_Materno;
        asigne.Departamental = item.Departamental;
        asigne.Placa = item.Placa;
        asigne.TransporteID = item.TransporteID;

        bool ret = true;

        if (item.Distribuciones.Count > 0)
        {
            foreach (var item3 in item.Distribuciones.OrderByDescending(c => c.Llegada))
            {
                asigne.Llegada = item3.Llegada;
                if (item3.Retorno == false)
                {
                    ret = false;
                }
            }
        }
        else
        {
            asigne.Llegada = item.Fecha_Creacion;
        }

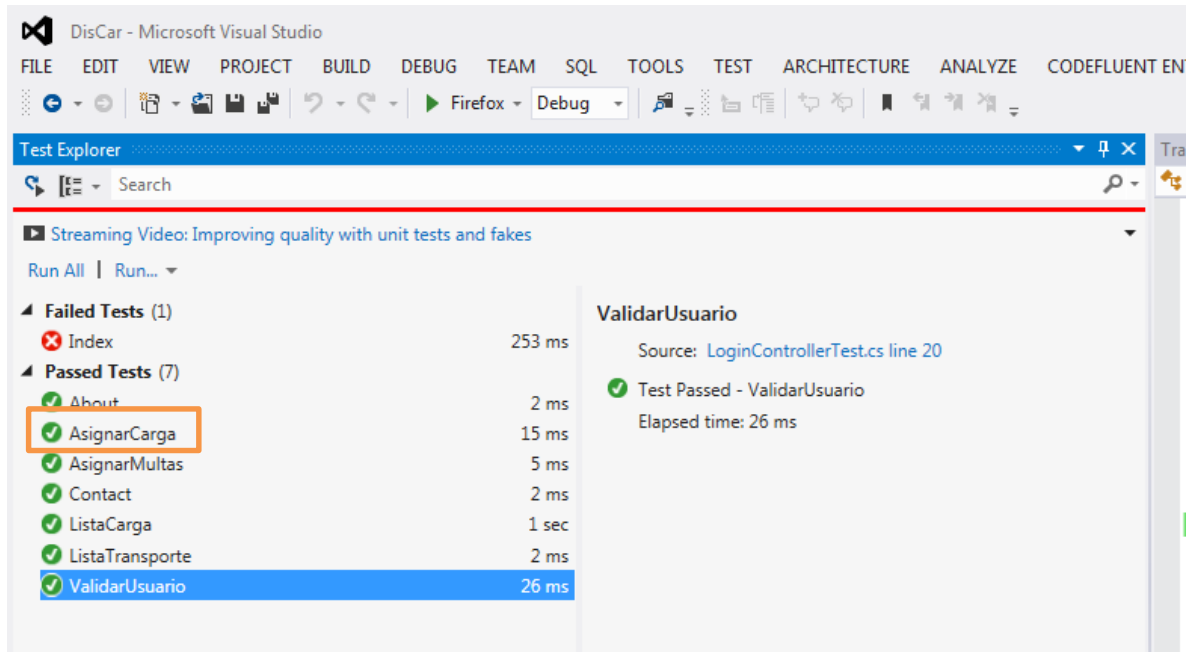
        // Assert
        Assert.IsNotNull(carga);
    }
}
```

**Fuente.** Sistema desarrollado

El resultado de la prueba. Después de realizar la prueba se puede observar la siguiente figura donde se tiene un resultado satisfactorio.

**Figura 3.37**

**Resultados de prueba unitaria del método asignar carga**



**Fuente.** Sistema desarrollado

### 3.5 Tercera Iteración monitoreo vehicular

En esta iteración se añadirán las funcionalidades necesarias para la gestión del monitoreo vehicular.

#### 3.5.1 Historia de usuario

##### 3.5.1.1 Historia de usuario Gestión de receptores

**Tabla 3.14**

**Historia de usuario gestión de receptores**

HISTORIA DE USUARIO	
Número:10	Usuario: Presidente de la Asociación
Nombre de la historia: Gestión de receptor	
Prioridad en el negocio: Alta	Riesgo de desarrollo: Alta

(Alta/media/Baja)	(Alta/media/Baja)
<b>Descripción:</b> Se requiere una interfaz donde se permita realizar las siguientes operaciones: Registro de datos de las características de los receptores que se implementaran en los vehículos en caso de cambio de chip se pueda editar el nuevo número del chip o caso contrario cambiar un receptor cuando tenga fallas.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.5.1.2 Historia de usuario control de monitoreo

**Tabla 3.15**

Historia de usuario control de monitoreo

HISTORIA DE USUARIO	
<b>Número:</b> 11	<b>Usuario:</b> Presidente de la asociación, y socios
<b>Nombre de la historia:</b> Control de monitoreo	
<b>Prioridad en el negocio:</b> Alta (Alta/media/Baja)	<b>Riesgo de desarrollo:</b> Alta (Alta/media/Baja)
<b>Descripción:</b> Se requiere implementar la interfaz del Google Maps para el proceso de monitoreo a los vehículos en tiempo real y gestionar algunas consultas de la misma.	
<b>Observaciones:</b> Ninguna	

**Fuente.** Elaboración propia

### 3.5.2 Planificación de entrega

**Tabla 3.16**

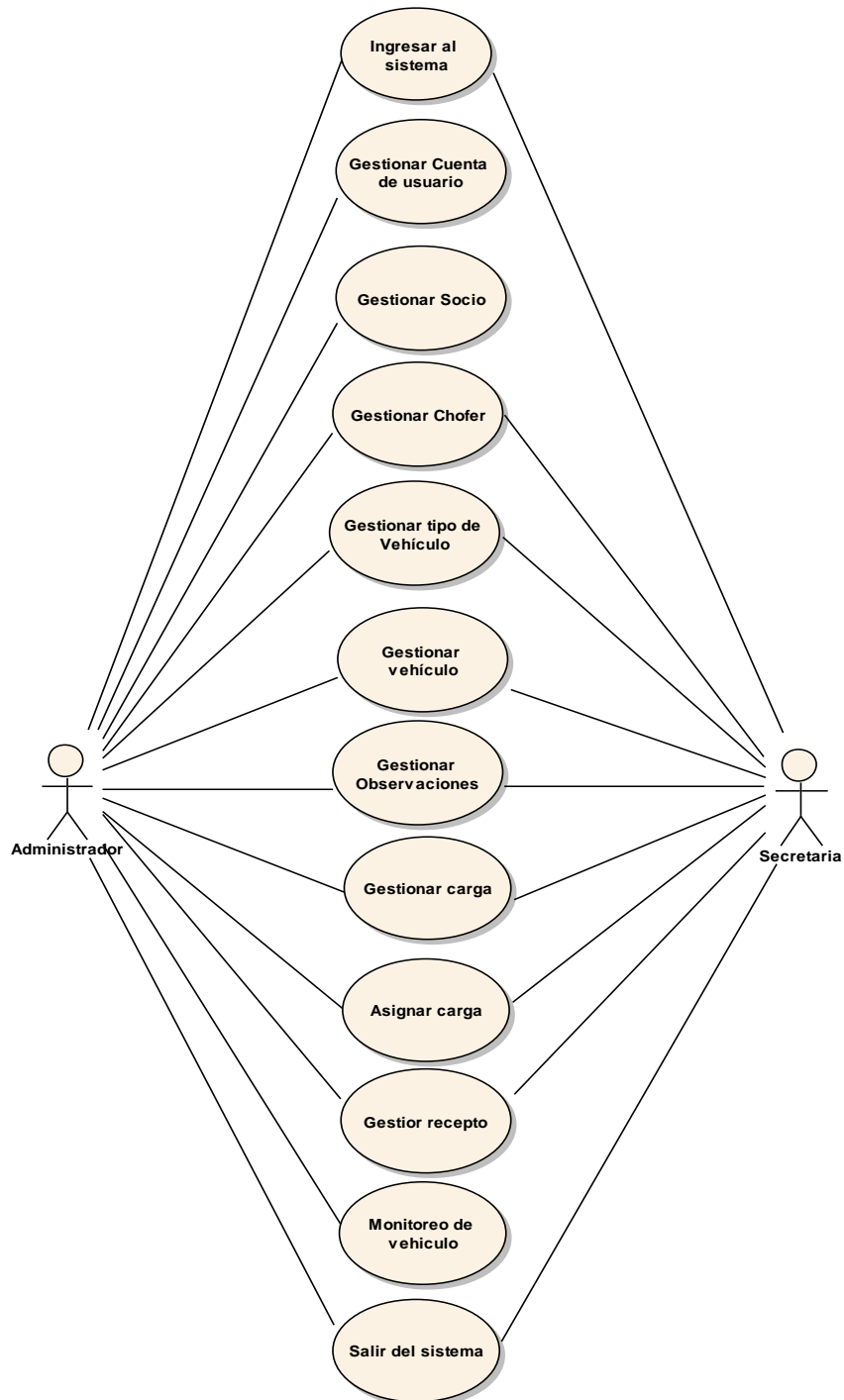
Estimación de esfuerzo de la tercera iteración.

Nº	ITERACIÓN	Nº	HISTORIAS DE USUARIOS	TIEMPO ESTIMADO		
				Sem.	Dia	Hrs
3	Monitoreo vehicular	10	Gestión de receptores	1	3	24
		11	Control de monitoreo	2	7	56
Tiempo total estimado				3	10	80

**Fuente.** Elaboración Propia

### 3.5.3 Diagrama de caso de uso General tercera iteración

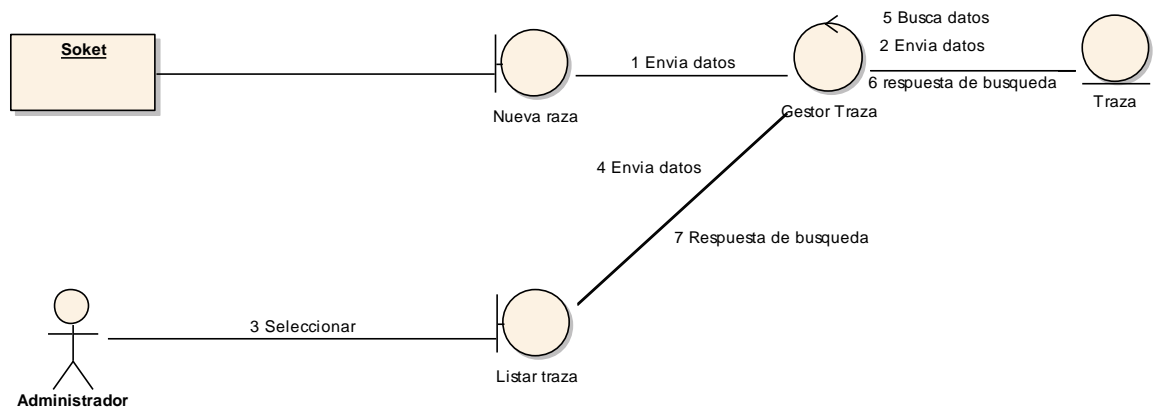
**Figura: 3.38**  
**Diagramas de caso de uso tercera iteración**



**Fuente.** Elaboración Propia

### 3.5.3.1 Diagramas de colaboración gestionar receptor

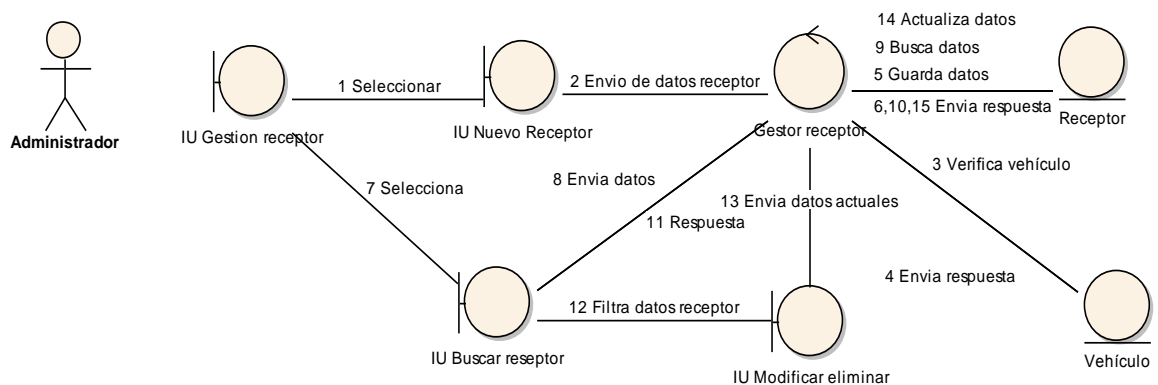
**Figura 3.39**  
**Diagrama de colaboración Gestionar Trazas**



**Fuente.** Elaboración propia

### 3.5.3.2 Diagrama de colaboración Asignación de carga

**Figura 3.40**  
**Diagrama de colaboración Gestionar Receptor**



**Fuente.** Elaboración propia

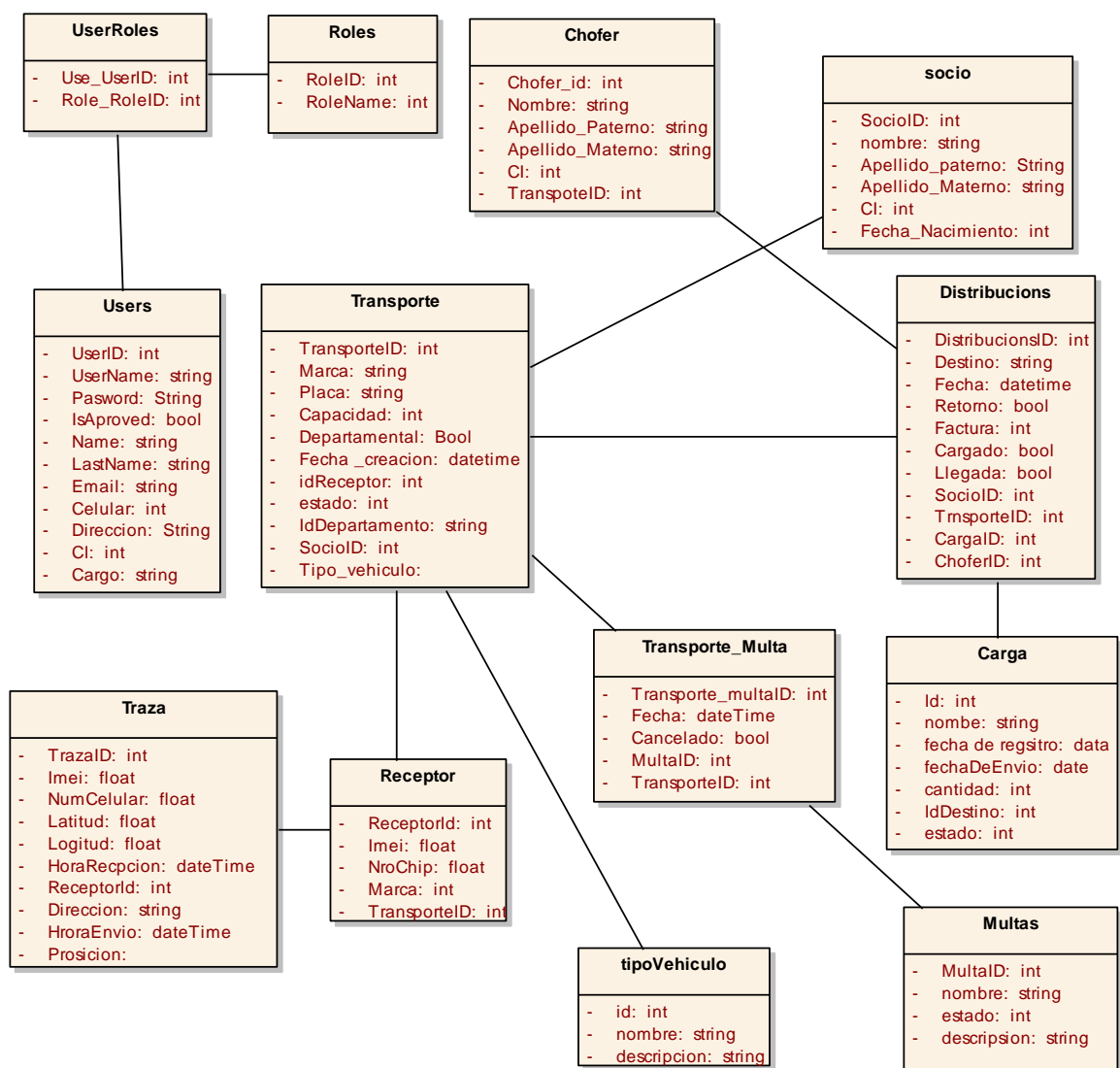
### 3.5.4 Diseño

Para el diseño se vio conveniente realizar un modelo de Entidad-Relación la cual ayudara para la creación de la Base de datos, de las clases y las vistas que se necesitaran.

A continuación en la siguiente figura se mostrará el modelo entidad - relación para la gestión de receptores y monitoreo de los vehículos.

**Figura 3.41**

**Modelo Entidad de relación tercera iteración**



**Fuente.** Elaboración propia.



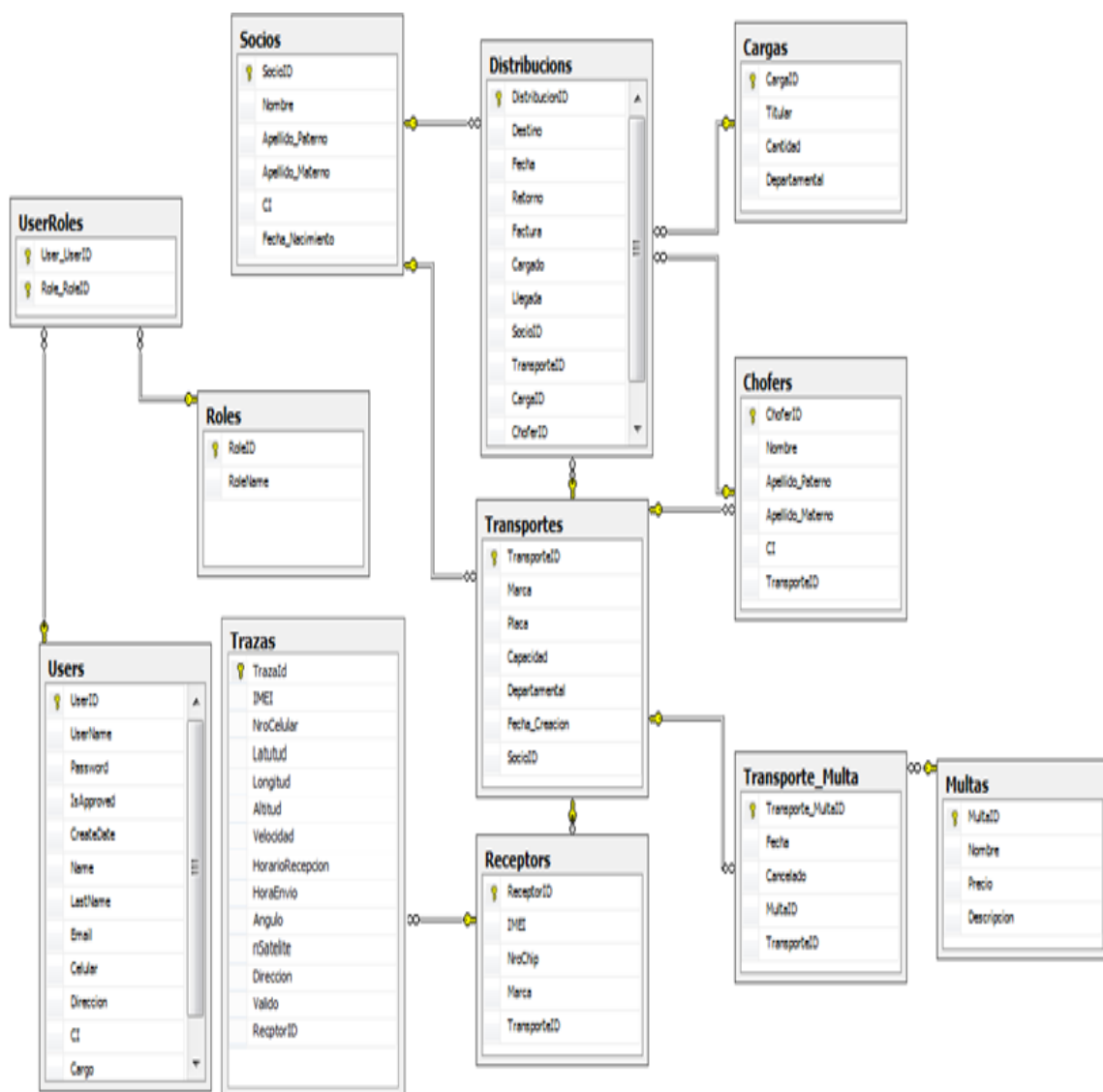
### 3.5.4.1 Diseño físico de la Base de datos de la tercera iteración

El diseño físico parte del esquema global obteniendo del modelo lógico, en él se obtiene la descripción de la implementación de la base de datos

A continuación en la siguiente figura se muestra el diseño físico del sistema que fue implementado en la base de datos.

**Figura 3.42**

#### **Diseño físico del sistema para la tercera iteración**



**Fuente.** Elaboración propia

### 3.5.5 Codificación

En la parte de la codificación se seguirá con la arquitectura seleccionada, en el proceso de gestión de monitoreo de vehículos.

#### 3.5.5.1 Modelo

En el modelo se define los atributo (datos) lo cual vemos en la siguiente figura.

**Figura 3.43**

**Modelo asignación de carga**

```
namespace DisCar.Models
{
    public class Traza
    {
        public int TrazaID { get; set; }

        public string IMEI { get; set; }
        public float NroCelular { get; set; }
        public float Latitud { get; set; }
        public float Longitud { get; set; }
        public float Altitud { get; set; }
        public decimal Velocidad { get; set; }
        public DateTime HoraRecepcion { get; set; }
        public DateTime HoraEnvio { get; set; }
        public int Angulo { get; set; }
        public float sSatelite { get; set; }
        public string Direccion { get; set; }
        public bool Valido { get; set; }

        public int ReceptorID { get; set; }

        public virtual Receptor Receptor { get; set; }
    }
}
```

**Fuente.** Sistema desarrollado

### 3.5.5.2 Controlador

El proceso que sigue cuando se realiza una petición de algún método es la consulta al controlador, a continuación se muestra el código de creación de receptores.

**Figura 3.44**  
**Controlador método crear receptor**

```
public ActionResult Create()
{
    ViewBag.TransporteID = new SelectList(db.Transportes, "TransporteID", "Marca");
    return View();
}

//
// POST: /Receptor/Create

[HttpPost]
public ActionResult Create(Receptor receptor)
{
    if (ModelState.IsValid)
    {
        db.Receptores.Add(receptor);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    ViewBag.TransporteID = new SelectList(db.Transportes, "TransporteID", "Placa", receptor.TransporteID);
    return View(receptor);
}
```

**Fuente.** Elaboración propia

### 3.5.5.3 Vista

Es la interfaz donde se puede comunicar al usuario con el sistema, para esta parte se tomara como ejemplo la interfaz de creación de receptores GPS.

**Figura 3.45**  
**Vista de Crear Receptor**

**ASOTRAM 1.0** Inicio Usuarios Transportes ▼ Asignación de Carga ▼ Reportes ▼

## Dispositivo GPS para Mario Liendre

### Asignar Receptor

---

IMEI

NroChip

Marca

Transporte

▼

[Regresar a la lista](#)

---

© ASOTRAM 2014

**Fuente.** Sistema desarrollado

**Figura 3.46**  
**Vista de lista de receptores**

## Lista de dispositivos GPS

IMEI	NroChip	Marca	Marca	
862170015466095	67689611	MEITRACK	Volvo	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>
45228637562191	68795689	MEITRACK	Nissan	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>
895632659878564	69865697	MEITRACK	Nissan	<a href="#">Editar</a>   <a href="#">Detalles</a>   <a href="#">Eliminar</a>

© ASOTRAM 2014

**Fuente.** Sistema desarrollado

### 3.5.6 Implementación del subsistema Socket

Los dispositivos GPS enviarán las trazas al servidor, para es necesario implementa un subsistema que escuche los datos enviados por los receptores.

Se implementa un subsistema llamada socket es la cual nos permitirá escuchar los datos enviados de los receptores a nuestro sistema.

### 3.5.7 Configuración del receptor GPS

En la siguiente figura se muestra la configuración del receptor GPS para el envío de la traza al sistema.

**Figura 3.47**  
**Vista Panel de configuración de receptor GPS**

Event	GPRS
SOS/Input 1 Active	<input checked="" type="checkbox"/>
Input 2 Active	<input checked="" type="checkbox"/>
Input 3 Active	<input checked="" type="checkbox"/>
Input 4 Active	<input checked="" type="checkbox"/>
Input 5 Active	<input checked="" type="checkbox"/>
SOS/Input 1 Inactive	<input checked="" type="checkbox"/>
Input 2 Inactive	<input checked="" type="checkbox"/>
Input 3 Inactive	<input checked="" type="checkbox"/>
Input 4 Inactive	<input checked="" type="checkbox"/>
Input 5 Inactive	<input checked="" type="checkbox"/>
Low Battery	<input checked="" type="checkbox"/>
Low External Power	<input checked="" type="checkbox"/>
Speeding	<input checked="" type="checkbox"/>
Enter Geo-fence	<input checked="" type="checkbox"/>
Exit Geo-fence	<input checked="" type="checkbox"/>
External Power On	<input checked="" type="checkbox"/>
External Power Off	<input checked="" type="checkbox"/>
No GPS Signal	<input checked="" type="checkbox"/>
Get GPS Signal	<input checked="" type="checkbox"/>
Enter Sleep	<input checked="" type="checkbox"/>
Exit Sleep	<input checked="" type="checkbox"/>
GPS Antenna Cut	<input checked="" type="checkbox"/>
Device Reboot	<input checked="" type="checkbox"/>
Heartbeat Report	<input checked="" type="checkbox"/>
Heading Change Report	<input checked="" type="checkbox"/>
Distance Interval Report	<input checked="" type="checkbox"/>

**Fuente.** Sistema desarrollado

En la siguiente figura muestra el código de configuración del socket en la cual configuramos la ip y el puerto que nos permitirá escuchar los datos.

**Figura 3.48**  
**Configuración de la IP y puerto en el socket**

```
<?xml version="1.0" encoding="utf-8"?>
] <configuration>
]   <appSettings>
     <add key="Secureway" value="Data Source=.\\sqlexpress;Initial Catalog=Asotram;Integrated Security=True" />
     <add key="logFilePath" value="LOG_TOTAL.log" />
     <add key="IP" value="200.58.80.74" />
     <add key="PUERTO" value="30175" />
     <add key="ClientSettingsProvider.ServiceUri" value="" />
   </appSettings>
]   <system.web>
]     <membership defaultProvider="ClientAuthenticationMembershipProvider">
]       <providers>
]         <add name="ClientAuthenticationMembershipProvider"
]           type="System.Web.ClientServices.Providers.ClientFormsAuthenticationMembershipProvider,
]             System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri="" />
]       </providers>
]     </membership>
]     <roleManager defaultProvider="ClientRoleProvider" enabled="true">
]       <providers>
]         <add name="ClientRoleProvider" type="System.Web.ClientServices.Providers.ClientRoleProvider,
]           System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri="" cacheTimeout="86400" />
]       </providers>
]     </roleManager>
]   </system.web>
] </configuration>
```

**Fuente.** Sistema desarrollado

En la siguiente figura muestra el código de almacenamiento de la traza en la base de datos.

**Figura 3.49**

### **Código de almacenamiento de la traza en la base de datos**

```
public class Program
{
    public void InsertarPosiciones(TCPListener.Entities.Program oPosiciones)
    {
        // TCPListener.DataAccess.ConnectionHelper oPosicionesCH = new ConnectionHelper(Common.CommonControls.GetConnectionString());
        string connectionString = @"Data Source=.\sqlexpress;Initial Catalog=Asotram;Integrated Security=True";

        try
        {
            string query = "INSERT INTO dbo.Traza (imei, chip, latitud, Longitud, Altitud, velocidad, FechaHora, idReceptor) " +
                "VALUES (@imei, @chip, @latitud, @longitud, @altitud, @velocidad, @fechaHora, @idReceptor) ";
            // create connection and command
            using (SqlConnection cn = new SqlConnection(connectionString))
            {
                using (SqlCommand cmd = new SqlCommand(query, cn))
                {
                    // define parameters and their values
                    cmd.Parameters.Add("@imei", SqlDbType.NVarChar, 50).Value = oPosiciones.Imei ;
                    cmd.Parameters.Add("@chip", SqlDbType.Int, 50).Value = oPosiciones.chip ;
                    cmd.Parameters.Add("@latitud", SqlDbType.Decimal, 50).Value = oPosiciones.Latitud ;
                    cmd.Parameters.Add("@longitud", SqlDbType.Decimal, 50).Value = oPosiciones.Longitud ;
                    cmd.Parameters.Add("@altitud", SqlDbType.Int).Value = oPosiciones.Altitud ;
                    cmd.Parameters.Add("@velocidad", SqlDbType.Decimal, 50).Value = oPosiciones.Velocidad ;
                    cmd.Parameters.Add("@fechaHora", SqlDbType.DateTime, 50).Value = oPosiciones.FechaHora ;
                    cmd.Parameters.Add("@idreceptor", SqlDbType.Int, 50).Value = oPosiciones.IdReceptor ;
                    // open connection, execute INSERT, close connection
                    cn.Open();
                    cmd.ExecuteNonQuery();
                    cn.Close();
                }
            }
        }
        catch
        {
            throw;
        }
        finally
        {
            }
        }
    }
}
```

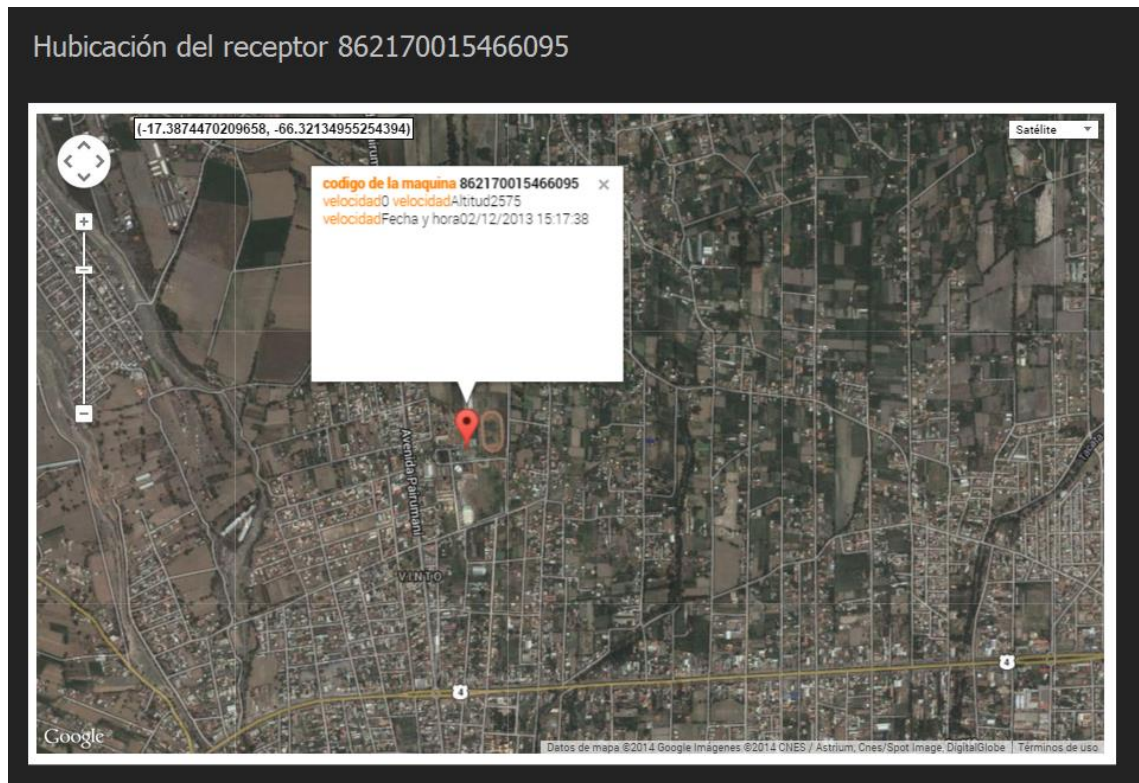
**Fuente.** Sistema desarrollado

#### **3.5.7.1 Implementación API de Google Maps en el sistema**

En la siguiente figura nos muestra la implementación del API de Google Maps en el sistema para la interfaz de las trazas en viada por el receptor al servidor.

**Figura 3.50**

**Implementación de Google Maps**



**Fuente.** Sistema desarrollado

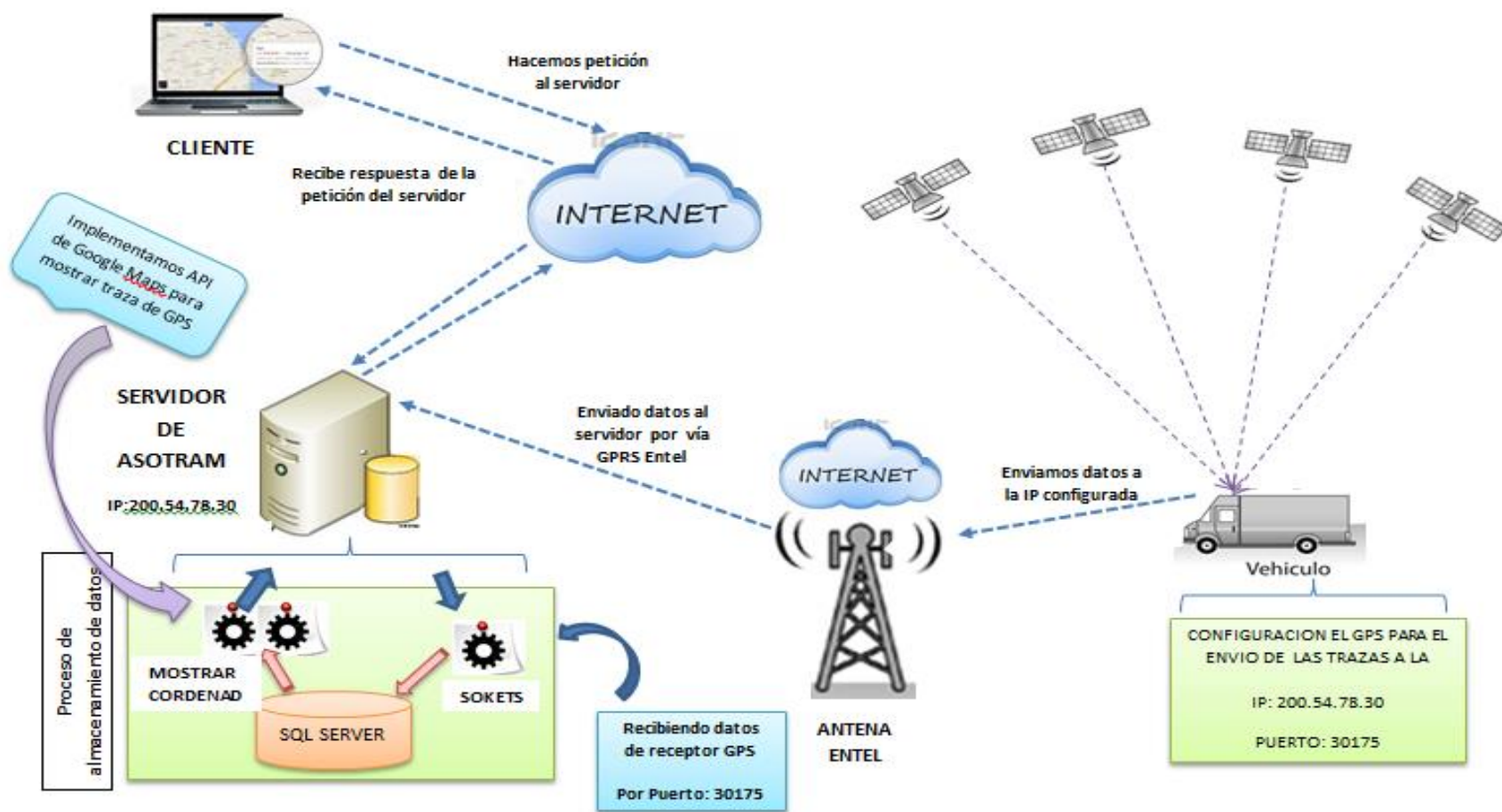
**3.5.7.2 Esquema General de la implementación de la tecnología GPS en el sistema**

En la siguiente figura se muestra el esquema general sobre la implementación de la tecnología GPS en el sistema, detallando así el proceso que se sigue.



Figura 3.51

Esquema de diseño de implementación y envío de datos del dispositivo GPS al servidor



Fuente. Elaboración propia

## CONCLUSIONES

Al concluir con el presente proyecto se resumen las siguientes conclusiones:

- Se elaboró el modelado de negocio alternativo, utilizando diagramas de flujo para el proceso de planificación y monitoreo en la Asociación de Transporte Mixto 24 de Septiembre para tener un amplio conocimiento de cómo es el proceso de planificación y monitoreo de vehículos.
- Con la implementación del subsistema de gestión de vehículos, socios y choferes se logró brindar una mejor administración de datos de los socios y vehículos.
- Se implementó mecanismos para el control del proceso de habilitación de vehículos permitiendo un eficiente control al momento de generar listas de vehículos habilitados.
- Se diseñó e implementó algoritmos para la optimización del proceso de asignación de cargas permitiendo, así conformidad y eficiencia al momento de la distribución de carga.
- Se implementó un subsistema Somet como puerta de entrada de los datos enviados por los receptores GPS al Sistema.
- Se implementó la tecnología GPS para el control y monitoreo de los vehículos.

## **RECOMENDACIONES**

Las recomendaciones nos ayudan a realizar adecuadamente, la manipulación del sistema para evitar errores de usuario.

- Se recomienda capacitar a los usuarios para tener un uso correcto del sistema
- Para tener un buen control sobre las infracciones y multas de los socios se recomienda desarrollar e implementa el sistema contable.
- Se recomienda adquirir o alquilar un servidor físico para tener un mejor rendimiento con la funcionalidad del sistema.
- Para la configuración del receptor se recomienda tener un experto capacitado en el tema.

## REFERENCIAS BIBLIOGRÁFICAS

- [1] Sistema de posicionamiento global. Disponible en: [http://www.serbi.ula.ve/serbiula/libros-electronicos/Libros/topografia\\_plana/pdf/CAP-10](http://www.serbi.ula.ve/serbiula/libros-electronicos/Libros/topografia_plana/pdf/CAP-10)  
Fecha 17 sep. 2013, a las 11:29
- [2] Paul Correia. "Control y política del GPS" en *Guía práctica del GPS*. Edición. Barcelona, España: Marcombo, 2002. Cap. 1. N°1.1 - 1.5. pp. 6-14.
- [3] Lawrence Lethan, "Receptores GPS" En *GPS fácil uso del sistema de posicionamiento global*. Primera edición. Barcelona, España: Paidotribo, 2002. Cap. 4. N°4.5. pp. 44.
- [4] M. Carmen España Boquera, "Servicio de acceso a la información" En *Servicios avanzados de telecomunicación*. Primera Edición. Madrid, España: Paidotribo, 2005. Cap. 7. N°7.4.3. pp. 154-160.
- [5] Cabana Cheung. (2013, Junio, 26). *MEITRACK MVT340 User Guide*. Disponible en: [http://www.meitrack.net/meitrack-support/manual/Personal-Vehicle-Motorcycle-manual/MEITRACK\\_MVT340\\_User\\_Guide\\_V2.8.pdf](http://www.meitrack.net/meitrack-support/manual/Personal-Vehicle-Motorcycle-manual/MEITRACK_MVT340_User_Guide_V2.8.pdf).
- [6] Roger S Pressman "El software en la Ingeniería de software" En *Ingeniería de Software un enfoque práctico*. 7ª Edición. D. F. México: McGraw-Hill, 2010. Cap. 1. N°1.3. pp. 7-10.
- [7] M. Rojas, A Orjuela "Las metodologías de desarrollo Ágil como una oportunidad para la ingeniería del software educativo", grupo de investigación CICOM, Colombia, versión final 24 de mayo 2008.
- [8] Vicenc Fernández "Modelado de procesos" En *Desarrollo de sistemas de información*. 1ra Edición. España Barcelona: UPC, 2006. Cap. 9. N°9.1. pp. 175-176.
- [9] Alfredo Weitzenfeld "Proceso de software" En *Ingeniería de Software Orientada a Objetos con UML, Java e internet*. 1ra Edición. España Madrid: THOMSON , 2005. Cap. 3. N°3.1. pp. 35.
- [10] HOWARD Podeswa "Análisis de procesos de negocio de extremo a extremo" En *programación UML*. 1ra Edición. España Madrid: ANAYA, 2010. Cap. 4. N°4.1. pp. 102.
- [11] FOWLER, Martin "¿Qué es UML?" En *UML gota a gota*. 1ra Edición. Mexico Naucalpan de Juarez: Wesley Longman , 2002. Cap. 1. N°1.1. pp. 1.
- [12] FOWLER, Martin (2004) 29 de Septiembre de 2010. "¿Ha muerto el diseño?" Disponible en: <http://www.programacionextrema.org/articulos/designdead.es>.

- [13] WEITZENFELD, Alfredo. *Ingeniería de software orientada a objetos con UML 2005*. México: Thomson Editores, S.A.
- [14] Stephen Walther. *ASP.NET MVC Framework Unleashed*. Estados Unidos de América. Pearson Educación S.A., 2010
- [15] Cesar de la torre LLORENTE “Arquitectura Marco N-Capas” En *Guia de arquitectura N-Capas orienta al dominio con .NET*. 1ra Edición. España: Krasis press, 2010. Cap. 3. N°1.1. pp. 33.
- [16] O. Santos García “Programación en C#” En *Microsoft Visual C# .NET*. 1ra Edición. España Madrid: McGraw-Hill, 2002. Cap. 1. N°1.1. pp. 1-2.
- [17] Fernando Berzal Galiano (2008, Marzo 25). La plataforma .NET. Disponible en: <http://elvex.ugr.es/decsai/csharp/dotnet/index>.
- [18] Ferguson, Jeff. (2003). *La Biblia de C#* Madrid: Ediciones Anayas Multimedia
- [19] Professional Visual Studio 2012, Bruce Jhonson, editorial Indianapolis, 2012, 15
- [20] Jeffrey Palermo. “High-Speed fundamentals” en *ASP .NET MVC4 IN ACTION*. Barcelona: Phil Haack. 2012, Cap. N° 1, 1.1, pp. 6.
- [21] OSORIO, Fray, “Bases de Datos Relaciones Teoría y Práctica”. Colombia: FONTO EDITORIAL ITM, 1ra Edición. 2008.
- [22] Julia Lerman. “Panorama general de la administración de base de datos” en *Programming Entity Framework*. Primera Edición. United State: O'Really. 2012, Cap. N° 1, 1.1, pp. 1.
- [23] Elvira Baidal, Vicente Santonja “Interfaz de programación en redes: los sockets” Curso de redes de computadores para ingenieros. 1ra Edición. España Valencia: editora UPV, 2005. Cap. 3. N°3.3. pp. 65-68.
- [24] Javier Areitio “Fundamentos de seguridad de la información” Seguridad de la información en redes, informática y sistemas de información. 1ra Edición. España: Paraninfo, 2008. Cap.1. N° 2.4. pp. 10-14.
- [25] Coral Calero y otros “Los nuevos modelos de ISO para la calidad y la calidad en uso del software” Calidad del producto y proceso de software. 1ra Edición. Madrid España: RA-MA, 2010. Cap.1. N° 2.4. pp. 10-14.

[26] Andrew Hunt. “Utilizar unit test” en *Pruebas unitarias pragmática en C#*. Segunda Edición. Biblioteca Pragmática. 2007, Cap. N° 2, pp. 26-59.