ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"ESTUDIO DEL FRAMEWORK NHIBERNATE Y SU APLICACIÓN EN EL SISTEMA PARA LA DETECCIÓN DE LLAMADAS EN LA COOPERATIVA TAXIPHONE"

TESIS DE GRADO

Previa obtención del Título de

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

IVON GABRIELA ZÚÑIGA LUNAVICTORIA

RIOBAMBA – ECUADOR
2010

AGRADECIMIENTO

Quiero expresar mis más sinceros agradecimientos a mi Dios por derramar bendiciones en mi vida, y darme fuerza, sabiduría, salud y la oportunidad de hacer éste sueño realidad.

A mis papitos Luis y Miriam por los consejos, amor, el apoyo incondicional brindado en mi vida y en el transcurso de mi carrera.

Un reconocimiento especial a mi Director de tesis, Ing. Wladimir Castro, que con su paciencia, guía contribuyó en gran parte el desarrollo y la culminación de ésta tesis.

A mi mejor amigo Iván, amigos, compañeros, al Ing. José Fabricio Rojas de la provincia de Trujillo - Perú que fue mi guía para el desarrollo de aplicación y a todos aquellos que de una u otra forma me han ayudaron hacer posible el alcance de éste objetivo.

DEDICATORIA

Este trabajo está dedicado a Dios, por la inspiración y fortaleza de todo momento.

A mis queridos padres Luis Zúñiga y Miriam Lunavictoria por brindarme el apoyo, la oportunidad la comprensión y la confianza para poder desarrollarme profesionalmente, por todas las enseñanzas, consejos y el buen ejemplo que me han dado.

Ivon Gabriela Zúñiga Lunavictoria.

FIRMAS RESPONSABLES

NOMBRE	FIRMA	FECHA
Ing. Iván Menes DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA	.....	.....
Ing. Raul Rosero DIRECTOR DE LA ESCUELA INGENIERÍA EN SISTEMAS	.....	.....
Ing. Wladimir Castro DIRECTOR DE TESIS	.....	.....
Ing. Iván Menés MIEMBRO DEL TRIBUNAL	.....	.....
Tlgo. Carlos Rodríguez DIRECTOR DEL CENTRO DE DOCUMENTACIÓN	.....	.....
NOTA DE LA TESIS	.....	

"Yo, Ivon Gabriela Zúñiga Lunavictoria, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la Tesis de Grado pertenece a la "**ESCUELA SUPERIOR POLITÉCNICA DEL CHIMBORAZO**".

……

Ivon Gabriela Zúñiga Lunavictoria.

ÍNDICE DE ABREVIATURAS

ADO	ActiveX Data Object Datos de objetos AcitveX
API	Application Programming Interface Interfaz de programación de aplicaciones
BCL	Base Class Library Biblioteca de clase base
BD	Data Base Base de Datos
CLR	Common Language Runtime Lenguaje común en tiempo de ejecución
CLS	Common Language Specification Especificación del Lenguaje Común
CRUD	Crear, Obtener, Actualizar Borrar Create, Retrieve, Update Delete
DAO	Data Access Object Objetos de acceso a Datos
DBMS	DataBase Management System Sistema Manejador de Base de Datos
GNU	General Public Licence Licencia Publica General
HBM	Hibernate Mapping Mapeo de Hibernate
HQL	Hibernate Query Language Lenguaje de Consultas de Hibernate
IEEE	Institute of Electronic and Electrical Engineers Instituto de ingenieros eléctricos y electrónicos
JIT	Just in Time Justo a Tiempo
LGPL	Lesser General Public Licence Licencia pública General Menor de GNU

MSF	Microsoft Solution Framework
MSIL	Microsoft Intermediate Language Lenguaje Intermedio de Microsoft
MVC	Model View Component Modelo Vista Componente
ODBC	Open DataBase Connectivity Conectividad abierta a base de datos
ORM	Object Relational Mapping Mapeo de Objeto-Rrelacional
POCO	Plain Object CLR Old Objetos planos viejos de CLR
POO	Programación orientada a objetos Object oriented programming
SQL	Structure Query Language Lenguaje de consultas estructurado
UML	Unified Modeling Language Lenguaje Unificado de Modelado
XML	Extensible Markup Language Lenguaje de marcas extensible

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El estudio de nuevas herramientas en el ámbito de desarrollo de sistemas cada día va innovando y a le vez proporciona ventajas en el momento de desarrollo, es el caso que nos brinda el nuevo framework de persistencia NHibernate, que inicialmente partió de la tecnología JAVA, migrando a la tecnología .NET, esto nos indica que cada vez más está evolucionando y proporcionando facilidades para el diseño, configuraciones ya sea de aplicaciones windows y web que se desarrollen

El principal objetivo del framework NHibernate, es proporcionar estabilidad, reducción de código y lo importante brinda la oportunidad de aprender y estrenar nuevas técnicas de programación aplicadas en JAVA a .NET, dicho de ésta manera se lo aplicará al sistema para la detección de llamadas en la cooperativa de taxis Taxiphone en una aplicación Windows bajo el lenguaje C#.NET 2005 y su base de datos en SQL 2005.

En el desarrollo de este proyecto se realiza un estudio del framework NHibernate, su estructura, composición, configuración, es así que lo hemos estructurado el contenido de la tesis en 5 Capítulos, el Capítulo I se relata los antecedentes, objetivos y justificación de la tesis, el Capítulo II se realiza el estudio y comparación entre Framework y ORM, el Capítulo III contiene el estudio directo del framework de persistencia NHibernate describiendo sus partes, instalación y configuración, en el Capítulo IV se encargara de la parte aplicativa en donde se detalla la metodología MSF en el que se desarrolla el sistema para la detección de llamadas y en el Capítulo V se desarrolla la comprobación de la hipótesis.

CAPITULO I

MARCO CONCEPTUAL

1.1. ANTECEDENTES

Hace un par de años se liberó la primera versión de un Framework de persistencia llamado NHibernate, una implementación para el Microsoft .Net Framework de Hibernate (implementado originalmente sólo para Java).

NHibernate es un Framework de persistencia, es decir provee herramientas que facilitan la tarea de persistir objetos. La motivación principal de NHibernate es abstraer por completo al desarrollador de la base de datos asociada al proyecto en desarrollo, es decir, el desarrollador debe pensar que sólo trabaja con objetos, los cuales puede guardar en una base de datos utilizando métodos de los mismos objetos, pero nunca escribir ni analizar una sentencia SQL en su código.

La arquitectura de NHibernate se puede representar de una manera simple como el siguiente diagrama:

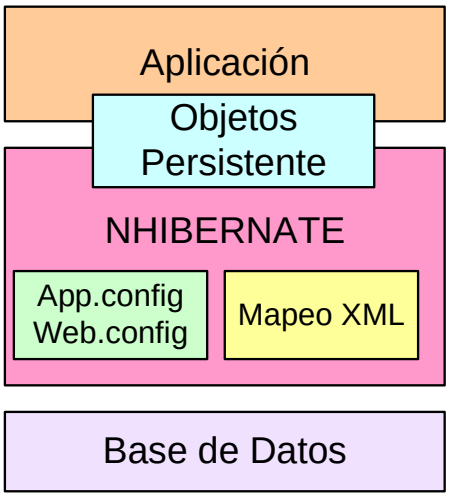


Figura Nº I. 1. Arquitectura General de NHibernate

En la Figura Nº I.1, podemos notar que la aplicación trabajará con objetos persistentes, pero sin comunicarse directamente con la base de datos. En su lugar, la comunicación será con el framework NHibernate, el cual se compone de una sección de configuración (puede ser archivo App.config o Web.config según nuestro proyecto sea Windows Forms o Web) y un conjunto de mapeos Objeto-Relacionales.

Utilizando estos elementos, NHibernate se comunicará con la base de datos y realizará las acciones requeridas por los objetos persistentes (inserción, actualización, borrado, selección).

Entrando un poco más en detalle, una arquitectura "ligera" de NHibernate es cuando la aplicación proporciona sus propias conexiones ADO.NET y maneja lo que son transacciones, entonces la arquitectura será como se ilustra en la Figura Nº I.2

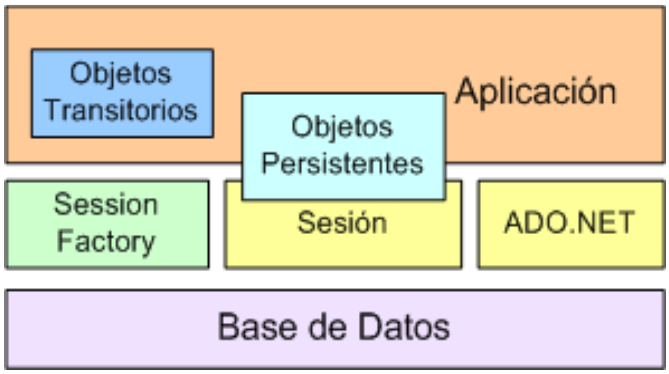


Figura Nº I. 2. Arquitectura ligera de NHibernate

Los objetos persistentes requieren almacenar estados, para esto es necesario utilizar una sesión (canal de comunicación entre la aplicación y la base de datos).

La sesión de comunicación será creada por una SessionFactory, que es un caché de los mapeos de una base de datos en particular. La SessionFactory puede ser configurada utilizando código o configurando los archivos App.config o Web.config.

Ahora, si deseamos utilizar todas las características que provee NHibernate, podemos utilizar una arquitectura completa como la que se muestra en la figura Nº I.3:

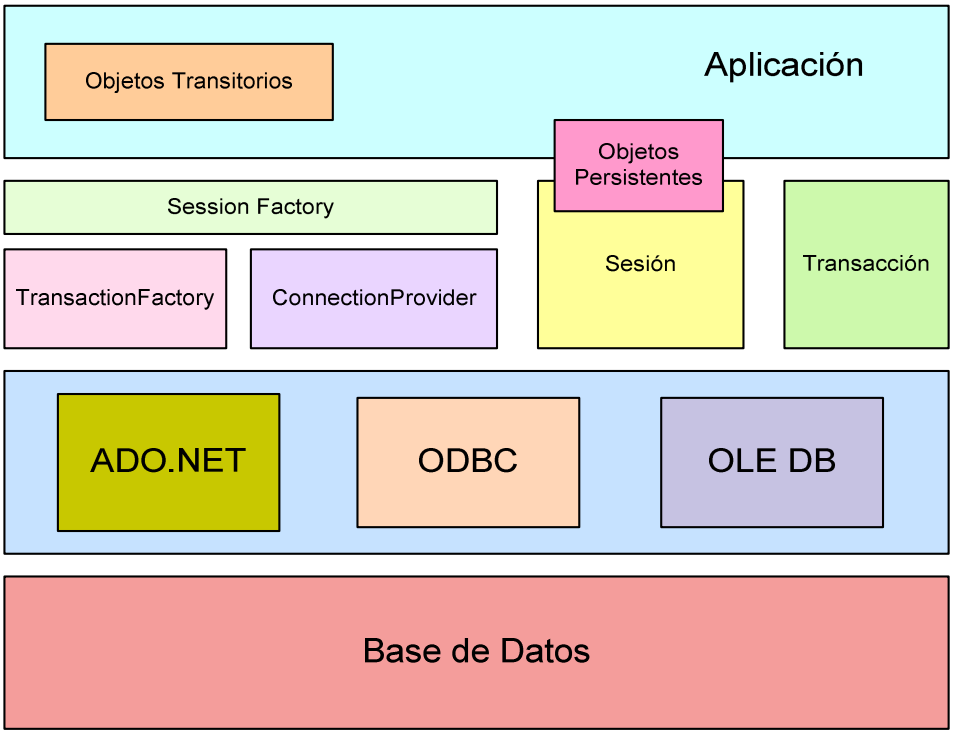


Figura Nº I. 3. Arquitectura completa de NHibernate

En esta arquitectura, NHibernate provee lo que es el control de transacciones (utilizando Transactions creadas por una TransactionFactory) y control de conexiones ADO.NET que no están expuestas a la aplicación, sin embargo pueden ser extendidas o implementadas por los desarrolladores.

En resumen, NHibernate es un Framework que pretende abstraer por completo al desarrollador de lo que es el manejo de persistencia en las aplicaciones. Al utilizar NHibernate el desarrollador sólo trabajará con objetos capaces de almacenar y recuperar estados, con todo lo que ello significa (manejo de relaciones entre objetos, jerarquía de objetos, etc.).

El ORM es un componente de software que me permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos (en este caso virtual). Debido a que lo standard es trabajar con BD relacionales, se deben realizar operaciones que permitan transformar un registro en objeto y viceversa. A esta funcionalidad se la llama Mapeo objeto-relacional (ORM).

Uno de los componentes ORM más utilizado es el Hibernate, surgido del ambiente Java y llevado al uso del framework .NET con la versión NHibernate

1.2. JUSTIFICACIÓN

Para la justificación de nuestro proyecto se tomará en consideración todos los aspectos relevantes que sean necesarios para obtener de mejor manera la justificación de nuestro proyecto

JUSTIFICACIÓN TEÓRICA

Importancia

NHibernate es la conversión de Hibernate de lenguaje Java a C# para su integración en la plataforma .NET. Al igual que muchas otras herramientas libres para esta plataforma, NHibernate también funciona en Mono.

Al usar NHibernate para el acceso a datos el desarrollador se asegura de que su aplicación es independiente en cuanto al motor de base de datos a utilizar en producción.

NHibernate es una poderosa e útil librería para .NET de persistencia de objetos para Bases de Datos Relacionales (DB2, Oracle, SQL Server, PostgresSQL, MySQL entre otros).Sólo se necesita cambiar una línea en el fichero de configuración para que podamos utilizar una base de datos distinta.

NHibernate es software libre, distribuido bajo los términos de la LGPL (Licencia Pública General Menor de GNU).

Beneficios

- ✓ Código abierto
- ✓ Soporta DataBinding
- ✓ Puede aceptar queries SQL directos (se los guarda en el xml)
- ✓ Se puede integrar con frameworks de MVC (como Struts para Java) o metaframeworks como Spring.
- ✓ Soporta las relaciones entre objetos (El usuario puede elegir el nombre y el tipo de datos de las llaves foráneas)
- ✓ Soporta agrupamiento (GROUP BY)
- ✓ Soporta agregación (COUNT, AVG, ETC.)
- ✓ Soporta llaves primarias compuestas

- ✓ Soporta asociaciones muchos a muchos y uno a muchos
- ✓ Soporta persistencia de propiedades a través de los campos de propiedades
- ✓ Soporta persistencia de propiedades a través de accessors (get/set methods or properties y pueden ser privados)
- ✓ Soporta trabajo offline y luego aplicar los cambios a la base de datos
- ✓ Soporta WebServices (la tecnología más moderna para la Integración de aplicaciones web, y el paradigma de programación moderno de programación orientada a servicios, publicación de servicios)
- ✓ Soporta tipos nulos
- ✓ No se requiere generar código pre compilado

Ventajas

- ✓ *Persistencia transparente:* Mis objetos del dominio no saben nada acerca de la base de datos donde son persistidos, el framework lo resuelve en forma automática utilizando archivos de mapping expresados en XML.
- ✓ *Soporte de polimorfismo:* Puedo cargar jerarquías de objetos en forma polimórfica.
- ✓ *Soporte de los 3 niveles de mapeo de herencia:* Puedo mapear toda una jerarquía de clases a una sola tabla, crear una tabla por cada clase concreta o crear una tabla por cada escalón de la jerarquía.
- ✓ *Soporte completo de asociaciones:* Los frameworks de ORM en especial el NHIBERNATE soportan el mapeo de todos los tipos de relaciones que pueden existir en un modelo de objetos del dominio (asociaciones 1..1, 1...N, N..M, unidireccionales y bidireccionales).

- ✓ *Soporte de carga de objetos Proxy:* Puedo cargar objetos que solo contengan la clave del objeto completo.
- ✓ *Soporte de caching:* En el contexto de una transacción, puedo disminuir la cantidad de veces que voy contra la base de datos cacheando en memoria los objetos que son accedidos varias veces.
- ✓ *Soporte de múltiples dialectos SQL:* Puedo independizarme completamente del tipo de base de datos utilizada. Mi aplicación puede persistir sus datos en SQL Server, en Oracle, en MySQL, etc. simplemente cambiando la configuración correspondiente.

JUSTIFICACIÓN METODOLÓGICA

En el estudio a realizar es necesario elegir la propuesta metodológica para el desarrollo de ésta investigación, la misma que nos permitirá guiarnos para el proceso de desarrollo e implementación de la aplicación.

Se aplicará la metodología SMF (Solution Microsoft Framework), siendo una metodología propia de Microsoft que se encuentra interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

- ✓ *Adaptable:* es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

- ✓ *Escalable:* puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- ✓ *Flexible:* es utilizada en el ambiente de desarrollo de cualquier cliente.
- ✓ *Tecnología Agnóstica*: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Metodología para el Desarrollo

Figura Nº I. 4. Métodos de Proceso de MSF

La figura Nº I.4, presenta el diagrama de la metodología con los procesos aplicables a sistemas informáticos

JUSTIFICACIÓN PRÁCTICA

El uso de nuevas herramientas de programación, permiten mejorar e incentivar a los programadores a mejorar e involucrarse más en éste ámbito.

Con el afán de ampliar los conocimientos hacia nueva herramientas tecnológicas de la programación, me incentivo a involucrarme en el uso y aplicación de la nueva tecnología que es el uso de ORM en la programación, siendo nuestro propósito aplicar los conocimientos adquiridos para aplicarlos en casos de la vida real.

Con los ORM, y el uso del framework NHIBERNATE, facilitará la aplicación de secuencias SQL en aplicaciones Windows o web, dependiendo el caso.

Es con el apoyo de la Cooperativa de Taxis TAXIPHONE que se ha planteado este reto, de forma tal que la Cooperativa tenga a disposición una aplicación Windows monousuario, que además se logré generar la detección de llamadas a través de la tarjeta de Fax módem del computador; y conlleve cubrir en su totalidad las necesidades de la cooperativa; en donde se permita integrar la información de los miembros de la unidades de taxis como de los usuarios que utilizan el servicio de la cooperativa; entre las funciones de la aplicación tenemos:

- ✓ Detección de las llamadas entrantes a la central de la cooperativa.
- ✓ Ingreso de cada usuario en donde consten los datos necesarios de ellos para su ubicación.

- ✓ Ingreso de los datos de cada unidad de taxi de la cooperativa.
- ✓ Generar reportes de todas las llamadas obtenidas en el día.
- ✓ Generar reportes de las llamadas atendidas con la información de la unidad de taxi.
- ✓ Generar reportes de las llamadas no atendidas.
- ✓ Generar reportes de las unidades de taxis en el día.

1.3. OBJETIVO GENERAL

Estudiar el Framework de persistencia NHIBERNATE y su aplicación en el sistema para la detección de llamadas en la Cooperativa de taxis TAXIPHONE

1.4. OBJETIVOS ESPECÍFICOS

- ✓ Estudiar los ORM y su relación con NHIBERNATE
- ✓ Realizar un estudio secuencial y cronológico de NHIBERNATE y su aplicabilidad en la vida real
- ✓ Establecer un patrón de diseño para la utilización de NHIBERNATE en Microsoft .NET 2005
- ✓ Implementar los NHIBERNATE como modelo ORM en un sistema base de datos
- ✓ Implantar una aplicación para la Cooperativa de Taxis TAXIPHONE usando los modelos estudiados

1.5. HIPÓTESIS

La aplicación del Framework de persistencia NHIBERNATE en el sistema de detección de llamadas mejorará en tiempo y seguridad la atención a los usuarios de la cooperativa TAXIPHONE

CAPITULO II

ESTUDIO DEL FRAMEWORK Y ORM

El estudio del framework y ORM, nos ayudará a distinguir a cada uno de ellos, para esto se realizará un estudio individual de cada uno de ellos explicando sus utilidades, ventajas, desventajas, estructuras y usos, siendo así que los framework se considera como el esqueleto de una estructura compuesta de varios componentes que facilitan al programador en su desarrollo de sistemas.

A diferencia de los ORM, que vienen a ser parte de la estructura de programación formando un puente de acceso a los datos de manera relacional a través de los objetos con las tablas y campos de la base de datos.

Tanto los framework como los ORM sirven como soporte para conseguir nuevas técnicas de programación de manera ordena que faciliten el desarrollo de sistemas informáticos indistintos del lenguaje de programación.

2.1. ESTUDIO DE FRAMEWORK

Se empezará dando un concepto generalizado de lo que significa “Framework”, partiremos de su traducción al español que es estructura, esqueleto, sistema; basándonos desde éste punto y tomando una visón en el ámbito de desarrollo se puede concebir una leve idea del objetivo de framework en términos generales, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar

Entonces se puede deducir que el framework de aplicación es una colección de componentes que colaboran entre sí para producir una arquitectura reutilizable en un conjunto de aplicaciones.

El concepto de framework se diferencia de una librería de clases tradicional en que los frameworks conforman el esqueleto de un particular dominio de las aplicaciones

Los frameworks son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional

Fuera de las aplicaciones en la informática, puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

No es más que una base de programación que atiende a sus descendientes manejado de una forma estructural y/o en cascada, posibilitando cualquier respuesta ante las necesidades de sus miembros, o secciones de una aplicación

Arquitectura

Dentro de este aspecto, podemos basarnos en el modelo MVC (Controlador - Modelo - Vista) ya que debemos fragmentar nuestra programación. Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema

Controlador.- Con este apartado podemos controlar el acceso a nuestra aplicación, esto pueden ser archivos, scripts o programas; cualquier tipo de información que permita la interfaz, así, podremos diversificar nuestro contenido de forma dinámica, y estática a la vez; pues, sólo debemos controlar ciertos aspectos

Modelo.- Este miembro del controlador maneja las operaciones lógicas y de manejo de información, cada miembro debe ser meticulosamente llamado, en su correcto nombre y en principio, con su verdadera naturaleza, el manejo de información, su complementación directa.

Vista.- Al final, a este miembro de la familia le corresponde dibujar, o expresar la última forma de los datos, la interfaz gráfica que interactúa con el usuario final del programa GUI, después de todo, a este miembro le toca evidenciar la información obtenida hasta hacerla llegar con el controlador.

Estructura

El modelo, al igual que el controlador y la vista, maneja todos los datos que se relacionen consigo solo es el proceso medio de la separación por capas que ofrece la arquitectura MVC, y sólo la vista, puede demostrar dicha información con lo cual ya hemos generado la jerarquía de nuestro programa Controlador, Modelo y Vista.

Lógica

Al parecer, debemos inyectar ciertos objetos dentro de sus parientes en esta aplicación, solo así compartirán herencia y coherencia en su aplicación.

2.1.1. FRAMEWORK .NET

.NET Framework es el corazón de la tecnología .NET. Es el marco de trabajo y ejecución común a toda la tecnología .NET. Es por lo tanto un elemento indispensable dentro de la tecnología .NET.

El .NET Framework es un componente de software que se instala en el sistema operativo. Podemos descargar el .NET Framework desde el sitio web de Microsoft:

- ✓ .NET Framework 1.1
- ✓ .NET Framework 2.0

Dentro del .NET framework están integrados los lenguajes .NET (C#, VB.NET, J#), el CRL, el CLS, ADO.NET, ASP.NET

.NET soporta varios lenguajes de programación, siendo los más populares:

- ✓ C#
- ✓ VB.NET
- ✓ C++ controlado
- ✓ J#
- ✓ Delphi

Todos ellos trabajan perfectamente integrados con el .NET framework, siendo C# el lenguaje principal, ya que es el único que ha sido desarrollado específicamente para .NET. Gran parte de la biblioteca de clases que componen el framework están escritas en C#.

.NET Framework es el conjunto de espacios de trabajo y clases que sirven de base a los lenguajes .NET. Es la evolución de las API de Windows y de la fundación de clases de Microsoft MFC hacia la completa encapsulación, cualquier programa que desarrollemos

utilizando la tecnología .NET, independientemente del lenguaje utilizado, estará basado en .NET Framework.

2.1.2. ESTRUCTURA Y COMPOSICIÓN DEL FRAMEWORK

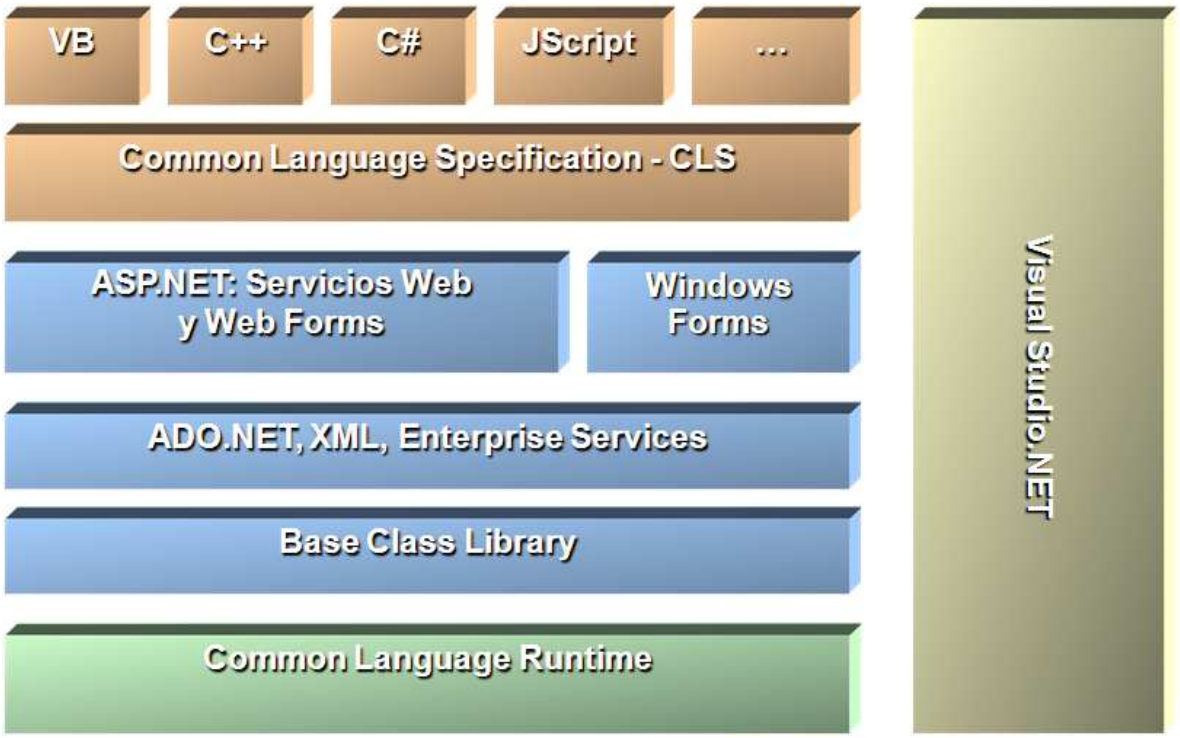


Figura Nº II. 5. Estructura del Framework .Net

En la figura II.5, se aprecia la estructura del Framework .Net , en donde tenemos una infraestructura de desarrollo que está compuesta por diversos recursos, entre los cuales se destaca el más importante, que es una máquina virtual conocida como CLR (Common Language Runtime), sobre la cual se ejecutan las aplicaciones. De este modo, nuestros programas ya no poseerán código nativo de ningún microprocesador en particular, sino instrucciones MSIL (Microsoft Intermediate Language) que serán

traducidas a código nativo en el momento de su ejecución por medio de un compilador JIT (Just In Time)

El framework también define una librería base de clases, BCL (Base Class Library), a la cual puede acceder cualquier desde lenguaje desarrollado para la plataforma. Por encima de la infraestructura se ubicará un conjunto de reglas básicas que debe implementar un lenguaje para poder ser parte de la familia .NET. Esta especificación es conocida como CLS (Common Language Specificeation). Finalmente, se encuentra el conjunto de lenguajes que cumplan con la especificación CLS, como el C#, el VB.NET, Managed C++, etc.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- ✓ Lenguajes de compilación
- ✓ Biblioteca de clases de .Net
- ✓ CLR (Common Language Runtime)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32. La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes

soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Language). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

2.2. MAPEO DE OBJETOS RELACIONALES ORM

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

Un ORM o Mapeador Objeto-Relacional propone una nueva forma de modelar los datos, y que permite solucionar la diferencia que existe entre los paradigmas de la POO (Programación Orientada a Objetos) y el modelo Relacional.

El modelo Relacional trata con relaciones y conjuntos por lo cual tiende a ser de carácter matemático, mientras que el modelo de la POO trata con objetos y las asociaciones entre ellos; el problema entre estos dos modelos surge en el momento de querer persistir los objetos de negocio.

El ORM debería resolver la mayoría de las cargas, permitiendo:

- ✓ **Mapear clases a tablas.-** propiedad a columna, clase a tabla.

- ✓ **Persistir objetos**.- A través de un método Orm.Save(objeto), encargándose de generar el SQL correspondiente.
- ✓ **Recuperar objetos persistidos**.- A través de un método objeto = Orm.Load(objeto.class, clave_primaria)
- ✓ **Recuperar una lista de objetos a partir de un lenguaje de consulta especial.-** A través de un método. ListObjetos = Orm.Find("Objeto FROM MyObject WHERE Objeto.Propiedad=5"), o algo más complejo ListObjetos = Orm.Find ("Objeto FROM MyObject WHERE Objeto.Relacion1.Relacion2.Propiedad2 = 5"), y el ORM transformará a través de varios joins de tablas.

Por ejemplo la escritura lo hacemos mediante ODBC que abre una conexión, crear una sentencia en SQL y copiar todos los valores de las propiedades de un objeto en la sentencia, ejecutarla y así almacenar el objeto, esto parece sencillo para un caso simple pero se vuelve complicado si el objeto posee muchas propiedades, o bien se necesita almacenar un objeto que a su vez posee una colección de otros elementos. Se necesita crear mucho más código, además del tedioso trabajo de creación de sentencias SQL.

Ahora si la lectura es a través de ODBC y se nos presenta el mismo problema que la escritura, a éste problema se conoce con el nombre de "Impedancia Objeto-Relacional" que es el conjunto de dificultades técnicas que surgen cuando una base de datos relacional se usa en conjunto con un programa escrito bajo POO.

Actualmente, las bases de datos relacionales solo pueden guardar datos primitivos, por lo que no podemos guardar objetos que vayamos creando en nuestra aplicación, sino

que lo que hacemos es convertir los datos del objeto en datos primitivos que si podremos almacenar en las tablas correspondientes de nuestras bases de datos. Si luego necesitamos ese objeto en alguna parte de nuestra aplicación, debemos de recuperar los datos primitivos de la base de datos y volver a construir el objeto.

El mapeo objeto-relacional lo que nos ayudará será precisamente a eso, a olvidarnos completamente de cómo convertir los objetos en datos primitivos para almacenarlos y viceversa.

El utilizar el ORM, nos puede proporcionar ciertas ventajas, pero como todo en esta vida, también nos proporcionará una serie de desventajas que a continuación se menciona.

Ventajas

- ✓ **Rapidez en el desarrollo.-** La mayoría de las herramientas actuales permiten la creación del modelo por medio del esquema de la base de datos, leyendo el esquema, nos crea el modelo adecuado.
- ✓ **Abstracción de la base de datos**.- Al utilizar un sistema ORM, lo que conseguidos es separarnos totalmente del sistema de base de datos que utilicemos, y así si en un futuro debemos de cambiar de motor de bases de datos, tendremos la seguridad de que este cambio no nos afectará a nuestro sistema, siendo el cambio más sencillo.
- ✓ **Reutilización**.- Nos permite utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.

- ✓ **Seguridad**.- Los ORM suelen implementar sistemas para evitar tipos de ataques como pueden ser los SQL injections.
- ✓ **Mantenimiento del código**.- Nos facilita el mantenimiento del código debido a la correcta ordenación de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.
- ✓ **Lenguaje propio para realizar las consultas**.- Estos sistemas de mapeo traen su propio lenguaje para hacer las consultas, lo que hace que los usuarios dejen de utilizar las sentencias SQL para que pasen a utilizar el lenguaje propio de cada herramienta.

Además de estas ventajas ofrece beneficios en el momento del desarrollo como:

- ✓ Trabaja con las clases diseñadas en el modelo del dominio
- ✓ No trabaja con filas de tablas, esto implica a que en el código no se emplea nada de DataRows ó RecordSets ó ResultSets
- ✓ Código orientado a objetos limpio, es decir que permite el uso de herencia, polimorfismo, composición en nuestras clases de negocio
- ✓ Permite crear, modificar, recuperar y borrar objetos persistentes, al recuperarlos nos permite navegar por las asociaciones entre objetos y luego actualizarlos al finalizar una transacción.
- ✓ Permite elegir la base de datos relacional subyacente con la que se desea interactuar como por ejemplo SqlServer, MySql, DB2, Oracle, etc.
- ✓ Genera automáticamente el código SQL usando una definición del mapeo objeto-relacional (XML mapping); ò el código de las clases

Desventajas

- ✓ **Tiempo utilizado en el aprendizaje**.- Este tipo de herramientas suelen ser complejas por lo que su correcta utilización lleva un tiempo que hay que emplear en ver el funcionamiento correcto y ver todo el partido que se le puede sacar.
- ✓ **Aplicaciones algo más lentas**.- Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá de transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos.

Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo), existiendo paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM.

En la programación orientada a objetos, las tareas de manejo de datos son implementadas generalmente por la manipulación de objetos, los cuales son casi siempre valores no escalares. Para ilustrarlo, considere el ejemplo de una entrada en una libreta de direcciones, que representa a una sola persona con cero o más números telefónicos y cero o más direcciones.

En una implementación orientada a objetos, esto puede ser modelado por un "objeto persona" con "campos" que almacenan los datos de dicha entrada: el nombre de la persona, una lista de números telefónicos y una lista de direcciones. La lista de números

telefónicos estaría compuesta por "objetos de números telefónicos" y así sucesivamente. La entrada de la libreta de direcciones es tratada como un valor único por el lenguaje de programación (puede ser referenciada por una sola variable, una instancia). Varios métodos pueden asociarse con el objeto, como uno que devuelva el número telefónico preferido, la dirección de su casa, etc.

Sin embargo, muchos productos populares de base de datos, como los productos SQL DBMS, solamente puede almacenar y manipular valores escalares como enteros y cadenas, organizados en tablas.

El programador debe convertir los valores de los objetos en grupos de valores simples para almacenarlos en la base de datos (y volverlos a convertir luego de recuperarlos de la base de datos), o solo usar valores escalares simples en el programa. El mapeo relacional de objetos es utilizado para implementar esta primera aproximación.

Utilidades del ORM en la programación

El mapeo objeto relacional ORM, es un enlace entre la programación orientada a objetos y el esquema relacional de la base de datos, como se ilustra en la Figura Nº II.6, muestra la relación y el acceso que brinda con la utilización correcta del ORM en donde minimiza los problemas de codificación repetitiva para acceder a la base de datos.

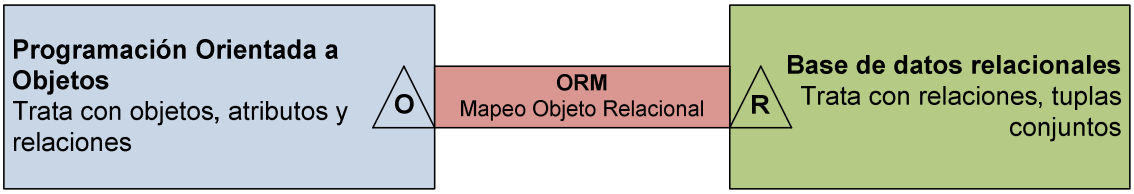


Figura Nº II. 6. Utilidades del ORM en la programación

Los ORM además de mapear permiten reducir susceptiblemente el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos, también proporcionan interfaces más simples para el manejo de objetos a través de su propio lenguaje de consulta, proveyendo al programador de configuraciones que le permiten optimizar los tiempos de respuesta en sus correspondientes aplicaciones

CAPITULO III

ESTUDIO DE FRAMEWORK DE PERSISTENCIA NHIBERNATE

El framework de persistencia NHibernate parte de del estudio, aplicabilidad en JAVA, a través del framework Hibernate, pero adaptado a la tecnología .NET; los dos framework poseen tienen el mismo fundamento y proceso de desarrollo, que gracias a las bondades que nos brinda, su estudio cada vez más se está ampliando y desempeñándose como soporte para los desarrolladores.

NHibernate no se basa en una sola estructura de desarrollo, más bien brinda alternativas de configuración y soporte que amplía el campo de desarrollo, con la utilización de sus componentes, capa de acceso a datos a través de sus archivos de mapeos configurables.

3.1. INTRODUCCIÓN A NHIBERNATE

Como se menciona en el capítulo I, en el apartado de los antecedentes; NHibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL, básicamente NHibernate es un framework de persistencia de mapeo objeto relacional que parte de Hibernate basado en la plataforma Java; que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Que NHibernate sea un framework de persistencia significa que tienen que pensar en NHibernate como algo que ejecuta queries SLQ (ejecuta insert, update, delete, select), NHibernate simplifica el acceso a datos y no debería tener mayores responsabilidades que eso.

Las técnicas de ORM nacieron exactamente para abstraer el acceso a la base de datos relacionales en una capa que resuelva el acceso a datos, nuestra aplicación será completamente orientada a objetos tanto en la capa de entidades de negocio y la capa de acceso a datos se ocupará de transformar esos objetos en instrucciones SQL para gestionar el estado persistente.

Como todas las herramientas de su tipo, NHibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación, entre la programación orientada a objetos POO y el usado en las bases de datos como es el modelo relacional; para lograr esto permite al desarrollador detallar cómo es su modelo

de datos, qué relaciones existen y qué forma tienen, con esta información, NHibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la POO. NHibernate convertirá los datos entre los tipos utilizados por .Net y los definidos por SQL. NHibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

NHibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente, también tiene la funcionalidad de crear la base de datos a partir de la información disponible.

NHibernate ofrece también un lenguaje de consulta de datos llamado **HQL** Hibernate Query Language, al mismo tiempo que una API para construir las consultas programáticamente conocidas como "criteria".

Interfaces de programación aplicaciones API, interactúan directamente con la capa de acceso a datos y estos con los archivos de configuración de nhibernate que poseen la extensión hbm que significa mapeos de hibernate ayudan a la interpretación de los métodos CRUD que se realiza en el desarrollo del sistema.

Para almacenar y recuperar objetos de la base de datos, el desarrollador debe mantener una conversación con el motor de NHibernate mediante un objeto especial, quizás el concepto clave más importante dentro NHibernate, que es la Sesión clase **Session**.

Un Session se puede equiparar a grandes rasgos al concepto de conexión de ODBC y cumple un papel muy parecido, es decir, sirve para delimitar una o varias operaciones relacionadas dentro de un proceso de negocio, demarcar una transacción y aporta algunos servicios adicionales.

La clase Session ofrece métodos como save(Object object), createQuery(String queryString), beginTransaction(), close(), etc. para interactuar con la BD, la clase que permite crear sesiones se denomina SessionFactory.

3.2. ARQUITECTURA DE LAS APLICACIONES NHIBERNATE

En la figura Nº III.7, se presenta la arquitectura del framework de persistencia NHibernate, en el que se establece cada componente que la conforma, empezamos por la primera capa en la que se encuentra el motor de la base de datos, en la segunda parte encontramos los componentes ADO que permiten tener acceso al motor de base de datos a través de la cadena de conexión representada por el Conection Provider, y para acceder a cada proceso, método u objeto se empleará los TransacctionFactory y sesiones representadas por los SessionFactory que son los que interactúan con las sesiones y transacciones para llegar a la aplicación.

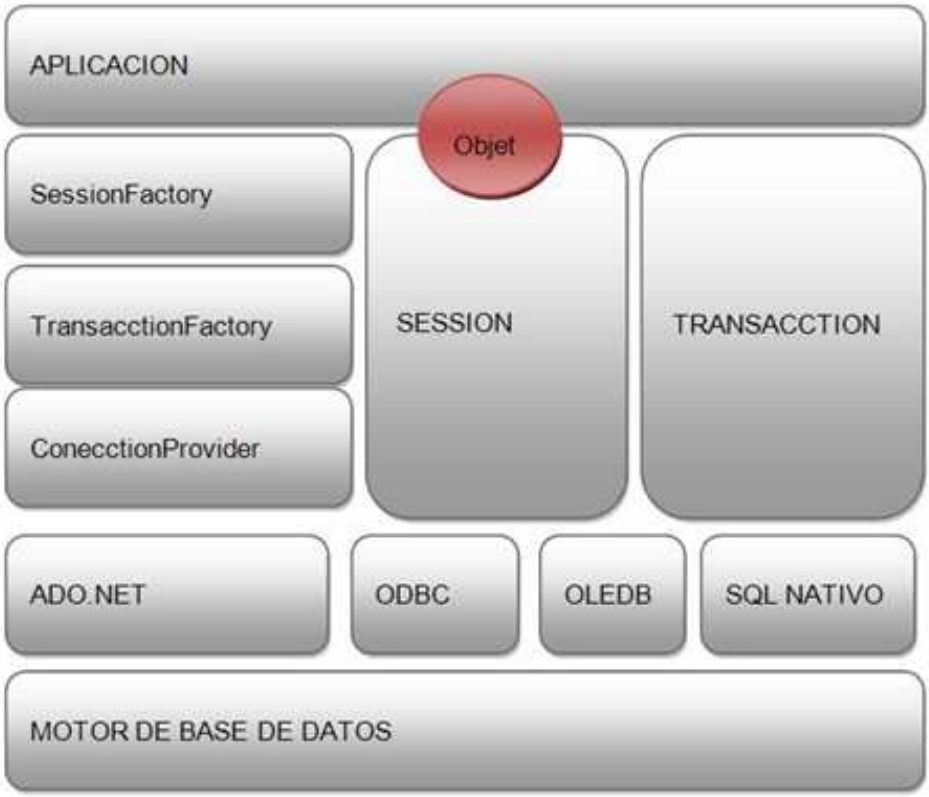


Figura Nº III. 7. Arquitectura de aplicaciones NHibernate

Se hará una breve descripción de los objetos que se están utilizando en la figura Nº III.7

ISession (NHibernate.ISession)

La interfaz ISession es la interfaz base para todas las sesiones. Contiene una propiedad ID( ) para identificar la sesión.

Una sesión es un contexto compartido entre todos los participantes en un intercambio de comunicación. Este contexto compartido puede caracterizarse de la siguiente manera:

- ✓ Establecimiento de la sesión: un proceso para establecer el contexto compartido entre todos los participantes.

- ✓ Correlación de mensajes: un método para poner en correlación un grupo de mensajes a una instancia de contexto compartido.

- ✓ La interfaz ISession no se informa acerca del número de participantes en una sesión o la duración de una sesión.

Estas propiedades de sesión se proporcionan si se asocia un identificador a una sesión.

ISessionFactory (NHibernate.ISessionFactory)

El objeto ISessionFactory utiliza la caché de asignaciones compilado para una única se de datos, es una fábrica de ISession y un cliente de IConnectionProvider, en el que puede tener un caché opcional de segundo nivel de los datos que se puede reutilizar entre las operaciones, en un proceso o un conjunto de nivel superior.

Persistentes Objetos y Colecciones

De vida corta, de un solo subproceso objetos que contengan el estado persistente y la función empresarial, estos pueden ser pocos ordinarios, la única cosa especial sobre ellos es que están actualmente asociados con exactamente uno ISession, tan pronto como se cierra la sesión, que será independiente y libre para usar en cualquier capa de

aplicación, por ejemplo, directamente como objetos de transferencia de datos hacia y desde la presentación.

ITransaction (NHibernate.ITransaction)

Un único subproceso, objeto de corta duración que utiliza la aplicación para especificar unidades atómicas de trabajo. Resúmenes de la aplicación subyacente transacción de ADO.NET. Un ISession podría abarcar varias ITransactions en algunos casos.

IConnectionProvider (NHibernate.Connection.IConnectionProvider)

Una fábrica para las conexiones de ADO.NET y los comandos. Resúmenes de aplicación de las implementaciones concretas específicos del proveedor de IDbConnection y IDbCommand, no exponer a la aplicación, pero puede ser extendido / implementado por el desarrollador.

Idriver (NHibernate.Driver.IDriver)

Una interfaz que encapsula las diferencias entre los proveedores de ADO.NET, como parámetro de convenciones de denominación y características compatibles con ADO.NET.

ITransactionFactory (NHibernate.Transaction.ITransactionFactory)

Una fábrica de instancias de ITransaction, no exponer a la aplicación, pero puede ser extendido / implementado por el desarrollador.

3.3. NÚCLEO DE INTERFACES

Las interfaces de programación son lo primero que tienen que aprender sobre NHibernate con el fin de utilizarlo en la capa de persistencia de la aplicación, uno de los objetivos principales del diseño de la API es mantener las interfaces entre componentes de software lo más estrecha posible, en la práctica, sin embargo, las API ORM no son especialmente pequeños.

La figura III.8, muestra las funciones de las interfaces NHibernate más importantes en el negocio y la capa de persistencia, en donde se presenta la capa de negocio por encima de la capa de persistencia, ya que la capa de negocios actúa como un cliente de la capa de persistencia en una aplicación tradicional por capas

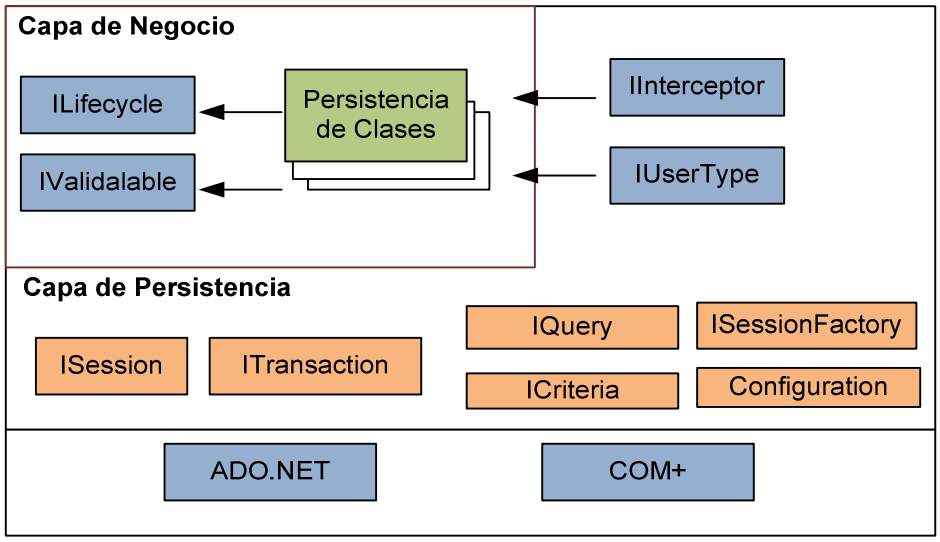


Figura Nº III. 8. Interfaces de NHibernate

Las interfaces llamadas por las aplicaciones para realizar operaciones CRUD básicas y las operaciones de consulta es decir; crear, seleccionar, actualizar y eliminar; éstas interfaces son el punto principal de la dependencia de aplicaciones empresariales y

lógica de control en NHibernate, éstas interfaces incluyen a ISession, ITransaction, IQuery y ICriteria.

Las interfaces llamada por el código de la aplicación para configurar la infraestructura de NHibernate, la más importante es la clase Configuration.

La devolución de llamada de interfaces que permiten la aplicación reaccionar a los acontecimientos ocurridos dentro de NHibernate, son las interfaces IInterceptor, ILifecycle y IValidatable.

Las interfaces que permiten la extensión de los servicios del mapeo de gran alcance NHibernate, como IUserType, ICompositeUserType y IIdentifierGenerator, estas interfaces son implementadas por código de la aplicación de la infraestructura si es necesario

NHibernate hace uso de las APIs. NET, incluyendo ADO.NET y su API ITransaction. ADO.NET proporciona un nivel rudimentario de la abstracción de la funcionalidad común a bases de datos relacionales, lo que permite casi cualquier base de datos con un controlador de ADO.NET con el apoyo de NHibernate.

Las cinco principales interfaces se enumeran a continuación y se utiliza en casi todas las aplicaciones de NHibernate, el uso de estas interfaces, puede almacenar y recuperar objetos persistentes y controlar debidamente las operaciones.

Interfaz ISession

La interfaz ISession es la interfaz primaria utilizada por las aplicaciones NHibernate, expone NHibernate métodos para encontrar, guardar, actualizar y eliminar objetos, una instancia de ISession es ligera y es barata de crear y destruir, esto es importante debido a que su aplicación tendrá que crear y destruir sesiones todo el tiempo.

La noción de una sesión de NHibernate es algo entre la conexión y la transacción, puede ser más fácil pensar en una sesión como un caché o colección de objetos cargados en relación con una sola unidad de trabajo, NHibernate puede detectar cambios en los objetos de esta unidad de trabajo a veces llame a la persistencia ISession porque también es la interfaz para las operaciones relacionadas con la persistencia, como el almacenamiento y recuperación de objetos

Interfaz ISessionFactory

La interfaz ISessionFactory no es ligera, está destinada a ser compartida entre los muchos hilos de aplicaciones, normalmente hay una sola instancia de ISessionFactory para el conjunto de aplicaciones creadas durante la inicialización de la aplicación, sin embargo, si su aplicación accede a bases de datos utilizando múltiples NHibernate, usted necesitará una SessionFactory para cada base de datos.

La interfaz SessionFactory también pueden almacenar los datos en caché que se ha leído en una unidad de trabajo, y que pueden ser reutilizados en una futura unidad de trabajo

o período de sesiones, esto es posible si se configura la clase y asignaciones de colección para utilizar el de segundo nivel de caché.

Interfaz de configuración

Las interfaz de configuración como su nombre lo indica se utiliza para configurar NHibernate, en donde la aplicación utiliza una instancia de configuración para especificar la ubicación del mapeo de los archivos y para establecer las propiedades específicas NHibernate antes de crear la interfaz ISessionFactory.

Interfaz ITransaction

La interfaz ITransaction es una opción de la API, las aplicaciones NHibernate pueden optar por no utilizar esta interfaz, en lugar gestionar las transacciones en su código de infraestructura propia que podría ser una transacción de ADO.NET o cualquier tipo de manual de operación que permite la aplicación para controlar límites de la transacción a través de una API consistente. Esto ayuda a mantener las aplicaciones portátiles NHibernate entre diferentes tipos de entornos de ejecución y contenedores.

Interfacces IQuery y ICriteria

La interfaz IQuery le ofrece potentes maneras de realizar consultas en la base de datos, al tiempo que controlar cómo se ejecuta la consulta, es la interfaz básica utilizada para

traer los datos utilizando NHibernate, las consultas están escritas en HQL o en el dialecto SQL nativo de tu base de datos

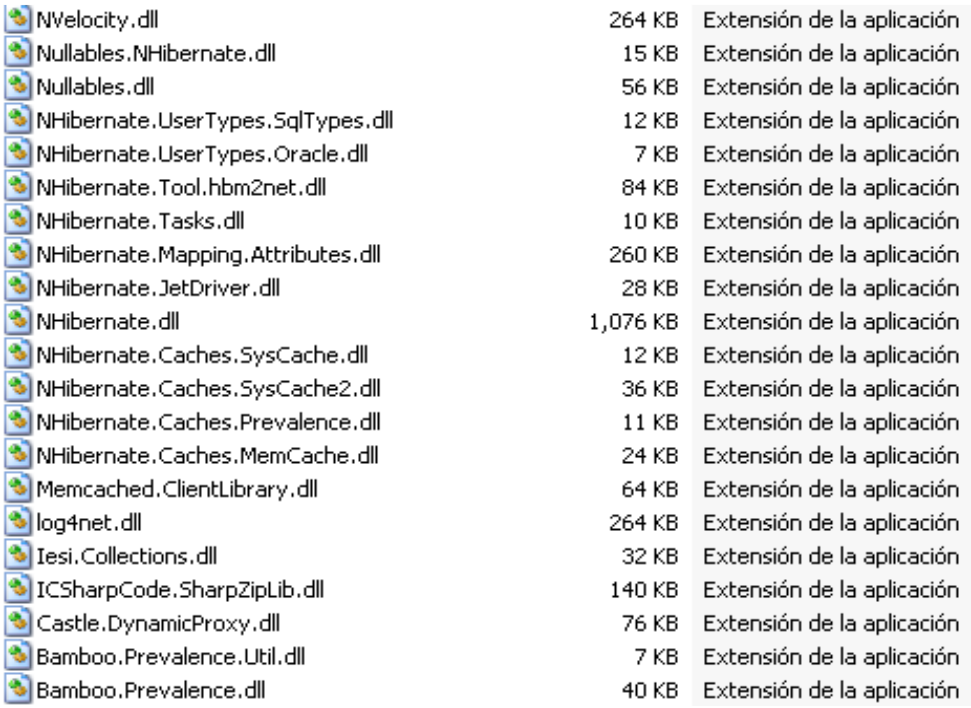
Una instancia IQuery es ligero y no se puede utilizar fuera de la ISession que lo creó, más bien se utiliza para obligar a los parámetros de consulta, limitar el número de resultados devueltos por la consulta y, finalmente, para ejecutar la consulta.

La interfaz es muy similar ICriteria, sino que le permite crear y ejecutar consultas criterios orientados a objetos.

Para ayudar a hacer menos detallado código de la aplicación, NHibernate proporciona algunos métodos útiles acceso directo en la interfaz ISession que le permiten activar una consulta en una sola línea de código.

3.4. CLASIFICACIÓN DE LAS LIBRERÍAS NHIBERNATE

Las principales librerías del framework NHibernate son por lo general 21, como se muestra en el gráfico III.9, en donde cada una están tiene su función especial según los requerimientos del desarrollador, las mismas que brindan soporte o accesibilidad a las funciones de NHibernate.



NVelocity.dll	264 KB	Extensión de la aplicación
Nullables.NHibernate.dll	15 KB	Extensión de la aplicación
Nullables.dll	56 KB	Extensión de la aplicación
NHibernate.UserTypes.SqlTypes.dll	12 KB	Extensión de la aplicación
NHibernate.UserTypes.Oracle.dll	7 KB	Extensión de la aplicación
NHibernate.Tool.hbm2net.dll	84 KB	Extensión de la aplicación
NHibernate.Tasks.dll	10 KB	Extensión de la aplicación
NHibernate.Mapping.Attributes.dll	260 KB	Extensión de la aplicación
NHibernate.JetDriver.dll	28 KB	Extensión de la aplicación
NHibernate.dll	1,076 KB	Extensión de la aplicación
NHibernate.Caches.SysCache.dll	12 KB	Extensión de la aplicación
NHibernate.Caches.SysCache2.dll	36 KB	Extensión de la aplicación
NHibernate.Caches.Prevalence.dll	11 KB	Extensión de la aplicación
NHibernate.Caches.MemCache.dll	24 KB	Extensión de la aplicación
Memcached.ClientLibrary.dll	64 KB	Extensión de la aplicación
log4net.dll	264 KB	Extensión de la aplicación
Iesi.Collections.dll	32 KB	Extensión de la aplicación
ICSharpCode.SharpZipLib.dll	140 KB	Extensión de la aplicación
Castle.DynamicProxy.dll	76 KB	Extensión de la aplicación
Bamboo.Prevalence.Util.dll	7 KB	Extensión de la aplicación
Bamboo.Prevalence.dll	40 KB	Extensión de la aplicación

Figura Nº III. 9. Librerías de NHibernate

Las librerías principales que se utilizan en los proyectos donde se trabajan con NHibernate, está conformada por la que lleva su mismo nombre NHibernate pues de ella partes otras librerías entre las principales están:

- ✓ NHibernate.Dialect
- ✓ NHibernate.Driver
- ✓ NHibernate.Cfg
- ✓ NHibernate
- ✓ NHibernate.Transacction

Vamos realizar una descripción de cada una de éstas librerías

NHibernate.Dialect

Las clases dentro del espacio de nombre NHibernate.Dialect, permiten establecer el dialecto para los diferentes motores de base de datos, en la Tabla Nº III.1 se presenta los principales dialectos.

Tabla Nº III.1. Dialectos de NHibernate

MOTOR DE BASE DE DATOS	DIALECTO
DB2	NHibernate.Dialect.DB2Dialect
MS SQL 2005	NHibernate.Dialect.MsSql2005Dialect
Oracle	NHibernate.Dialect.OracleDialect
MySQL 3.0- 4.0	NHibernate.Dialect.MySQLDialect
PostgreSQL	NHibernate.Dialect.PostgreSQLDialect

NHibernate.Driver

Las clases dentro de este espacio de nombre nos proveen el driver de conexión a los diferentes motores de base de datos relaciones, estos no son los únicos, pero si considero los más importantes, usando estos driver en nuestro archivo de configuración se lo instancia como clases dentro de nuestro proyecto .net para poder setear sus propiedades vía código, en la Tabla III.2. se muestra los principales driver de NHibernate

Tabla Nº III.2. Driver de conexión de NHibernate

MOTOR DE BASE DE DATOS	DRIVER
DB2	NHibernate.Driver.DB2Driver
MS SQL 2005	NHibernate.Driver.SqlClientDriver
Oracle	NHibernate.Driver.OracleClientDriver
MySQL 3.0- 4.0	NHibernate.Driver.MySqlDataDriver
PostgreSQL	NHibernate.Driver.NpgsqlDriver

NHibernate.Cfg

La clase **Configuration** del Espacio de Nombre NHibernate.Cfg

Configure([“string”]).- Lee el archivo de configuración: hibernate.hbm.xml, el cual contiene las propiedades para configurar el framework de acuerdo al motor de base de datos.

La Clase **ISessionFactory** del Espacio de Nombre NHibernate.Cfg:

BuildSessionFactory().- Crea una sesión que es la abstracción de una conexión a una base de datos.

NHibernate

La Clase **ISession** del Espacio de Nombre NHibernate abre una sesión que es una abstracción de una conexión al Motor de Base de Datos

***Métodos Principales*:**

- ✓ **Save([object])**.- Nos permite Guardar un objeto , para el Motor de Base de datos se trata de una inserción de un registro.
- ✓ **Update([object]).-** Nos permite actualizar un objeto , por el valor de referencia del objeto , o también a través de su propiedad definida en el Archivo de Mapeo XML ("id")
- ✓ **Delete([object]).-** Nos permite eliminar un objeto y eliminarlo de la Base de Datos, a través objeto por referencia , o a través de la propiedad "id"

La Clase **ICriteria** del Espacio de Nombre NHibernate.- Nos ayuda a Obtener la Información:

CreateCritertia([object]).- De una session determinada Listo los tipo Factura, a través del Metodo List<T> de Tipo Factura

Ejm: sesion.CreateCriteria(typeof (Factura)).List<Factura>();

NHibernate.Transacction

La Clase **ITransacction** del Espacio de Nombre de NHibernate.Transacction, utiliza los siguientes procesos con los cuales interactúa.

- ✓ **BeginTransaction:** Para una determinada Session, es la que permite empezar una Transacción. Ejm:

```
ITransaction transaccion = null
transaccion = sesion.BeginTransaction()
```

- ✓ **Commit( ):** Guarda todos los cambios en la session, una vez que la transacción ha terminado.
- ✓ **Rollback( ):** Si la session no se va llevado de forma correcta , no guarda nada.

3.5. PERSISTENCIA DE OBJETOS

La persistencia es la capacidad de un lenguaje de programación o entorno de desarrollo para almacenar y recuperar el estado de los objetos de forma que sobrevivan a los procesos que las manipulan, dicho de otra manera es conseguir que los datos de un objeto sobrevivan a la ejecución de un proceso padre (en donde fueron creados), para que puedan ser reutilizados por otros procesos, entonces se hace referencia al término de persistencia a la capacidad de un elemento para hacerse persistente, es decir, a la capacidad de poder almacenar el estado de un elemento de modo que sea posible recuperarlo posteriormente

Una instancia persistente es aquella cuyos datos perduran a la ejecución del proceso que materializó la instancia. Una instancia transitoria o temporal, es toda instancia cuyos datos desaparecen cuando finalizan los procesos que la manipulan, en ambos casos, las instancias en sí, como estructuras de datos residentes en memoria, desaparecen al finalizar los procesos que las crearon.

Veámoslo con un sencillo ejemplo, imaginemos la ejecución de un programa que solicita introducir nuestro nombre que será usado repetidas veces en distintas operaciones. Si este dato es recogido en una instancia transitoria, cuando finalice el programa y lo volvamos a ejecutar, deberemos nuevamente introducir el dato; pero si está asociado a una instancia persistente, el dato introducido podría ser recuperado y mostrado en sucesivas ejecuciones del programa.

De otra parte, los servicios de motor de base de datos orientada a objetos ODBMS ofrecen servicios de persistencia, que adoptan más de las siguientes aproximaciones

- ✓ **POR TIPO**.- Un objeto puede llegar a ser persistente cuando es creado de algún tipo (clase) dotado de la capacidad de persistir o de un subtipo (clase descendiente) de estos. Los tipos persistentes son distintos de los tipos cuyas instancias son transitorias. El tipo podría ser identificado como persistente en su declaración, o por herencia de algunos de los tipos predeterminados del sistema, de forma parecida, la serialización obliga que las clases implementen la interfaz Serializable.

- ✓ **POR INVOCACIÓN EXPLICITA**.- El programador invoca un método que provoca que un objeto pase a ser persistente. La llamada al método puede ser en la creación del objeto o en cualquier instante, según la implementación
- ✓ **POR REFERENCIA**.- un objeto es hecho persistente por otro que es persistente. Añadir un objeto a una colección persistente, la asignación a cierto tipo de referencias, provocan que un objeto transitorio pase a ser persistente

En la figura Nº III.10 se presenta un ejemplo empleando la persistencia Por Tipo y Por Referencia, en cambio en la figura Nº III.11 se ilustra un ejemplo aplicando la técnica de persistencia Por Invocación y por Referencia

```
private void persisteMetodoODMGModos(Cliente cli) {

  Database db = Database.open("OODBMSBANCO", ......);
  Transaction tr = new Transaction();
  try {
    tr.begin();
    ListOfObject listaClientes =
                        (ListOfObject) db.lookup("Clientes");
    if (listaClientes == null) {
       // Por Tipo persistente;
      listaClientes = new ListOfObject(db);
      db.bind(listClientes, "Clientes");
    }
    // por Referencia
    listaClientes.add(Cli);
    tr.commit();
    db.close();
  }
  catch (ODMGException e) {
    //.....
```

Por tipo

Por Referencia

Figura Nº III. 10. Ejemplo de Persistencia por Tipo y por Referencia

```
 private void persisteMetodoJDOModos(Cliente cli) {
   PersistenceManagerFactory pmf =
       JDOHelper.getPersistenceManagerFactory(properties);
   PersistenceManager pm = pmf.getPersistenceManager();
   / ....
       Transaction tr = pm.currentTransaction();
   try {
     tr.begin();
     todosLosClientes = pm.getExtent(LisOfObject.class, true);
     // ....
     if (todosLosClientes == null) {
       //por invocación
       pm.makePersistent(listaClientes);
     .......
     }
     //....
     // por Referencia
     listaClientes.add(Cli);
     tr.commit();
     pm.close();
   }
   catch (JDOException e) {
     //.....
```

Por Invocación

Por referencia

Figura Nº III. 11. Ejemplo de Persistencia por Invocación y por Referencia

3.6. MOTOR DE PERSISTENCIA

Un motor de persistencia es un sistema de desarrollo, también conocido como framework o librería, que nos permite hacer persistentes los elementos de nuestras aplicaciones o modelos, estos elementos pueden ser objetos, tipos primitivos de datos, arrays, etc., mientras que la persistencia se puede realizar sobre múltiples destinos como bases de datos relacionales, bases de datos orientadas a objetos, sistemas legacy, etc.

Los motores de persistencia no sólo nos permiten hacer persistentes nuestros objetos, sino que también nos permiten realizar consultas complejas que nos ofrezcan jerarquías completas de objetos, realizan optimizaciones automáticamente, resuelven asociaciones y dependencias, etc., la ventaja de utilizar un motor de persistencia es la sencillez del modelo de programación que promueven.

Un programador ya no se tiene que preocupar de realizar consultas manuales con ODBC y simplemente utilizará las funciones que le ofrece el motor; éstas funciones además de realizar múltiples optimizaciones de rendimiento le garantizan que la aplicación será compatible con todas las bases de datos que soporte el motor de persistencia que utilice el programador y precisamente uno de los motores de persistencia más utilizado y activo en su desarrollo es NHibernate

En la Figura Nº III.12, se esquematiza en funcionamiento de un motor de persistencia el que interactúa con una base de datos y el programa

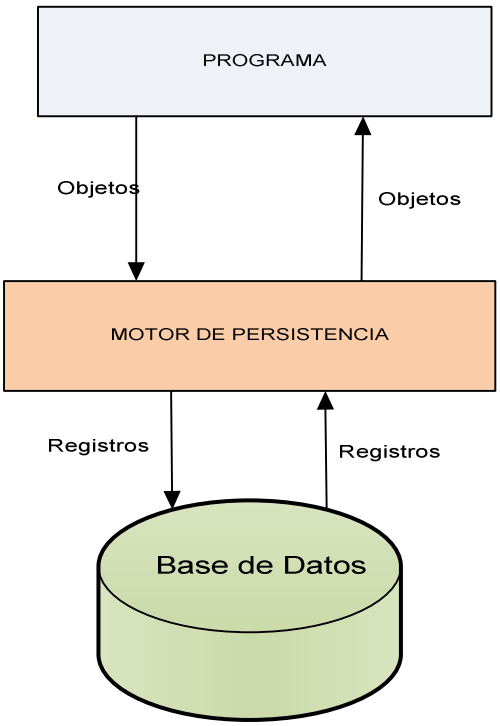


Figura Nº III. 12. Motor de Persistencia

El motor de persistencia traduce entre los dos formatos de datos: de registros a objetos y de objetos a registros, la situación se ejemplifica en la figura III.12, cuando el programa quiere grabar un objeto llama al motor de persistencia, que traduce el objeto a registros y llama a la base de datos para que guarde estos registros; de la misma manera, cuando el programa quiere recuperar un objeto, la base de datos recupera los registros correspondientes, los cuales son traducidos en formato de objeto por el motor de persistencia.

El programa sólo ve que puede guardar objetos y recuperar objetos, como si estuviera programado para una base de datos orientada a objetos; la base de datos sólo ve que guarda registros y recupera registros, como si el programa estuviera dirigiéndose a ella de forma relacional, es decir, cada uno de los dos componentes trabaja con el formato

de datos “idioma” que le resulta más natural y es el motor de persistencia el que actúa de traductor entre los dos modelos, permitiendo que los dos componentes se comuniquen y trabajen conjuntamente.

Esta solución goza de las mejores ventajas de los dos modelos.

- ✓ Brinda la capacidad de programar con orientación a objetos, aprovechando las ventajas de flexibilidad, mantenimiento y reusabilidad.
- ✓ Se puede usar una base de datos relacional, aprovechándonos de su madurez y su estandarización así como de las herramientas relacionales que hay para ella.
- ✓ El programa solo ve que puede guardar y recuperar objetos, como si estuviera programado para una base de datos orientada a objetos y a su vez la base de datos sólo ve que guarda registros y recupera registros como si el programa estuviera dirigiéndose a ella de forma relacional

.

Se calcula que un motor de persistencia puede reducir el código de una aplicación en un 40%, haciéndola menos costosa de desarrollar, además, el código que se obtiene programando de esta manera es más limpio y sencillo, y por lo tanto, más fácil de mantener y más robusto: por añadidura, el motor de persistencia no sólo simplifica la programación, sino que permite hacer ciertas optimizaciones de rendimiento que serían difíciles de programar por nosotros mismos.

Es un componente de software que realiza la traducción de registros de base de datos relacionales a objetos y viceversa. Esta traducción es conocida como mapeo o mapping

que precisamente es una técnica de programación para convertir el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en la base de datos relacional.

3.7. TRABAJANDO CON NHIBERNATE

Tomando en consideración de que NHibernate es un framework para .Net, el cual facilita el mapeo de atributos mediante archivos XML que permiten establecer relaciones entre una base de datos relacional y objetos .NET, creando un entorno Objeto-Relacional.

Para lograr esto es necesario detallar cómo es el modelo de datos, qué relaciones existen y qué forma tienen; con esta información NHibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos, NHibernate convierte los datos entre los tipos utilizados por .NET y los definidos por SQL.

NHibernate genera las sentencias SQL y libera al programador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todas o por lo menos la mayoría de las bases de datos con un ligero incremento en el tiempo de ejecución, lo que es una ventaja importe, nunca se está atado a usar siempre un mismo manejador de base de datos.

NHibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También ofrece un

lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente conocida como "criteria".

Hablando un poco de la composición de NHibernate, pueden destacar 3 módulos:

1. **Proveedor de Conexiones**.- Este administrador provee un manejo eficiente de las conexiones a las base de datos; las conexiones son la parte más costosa al momento de interactuar con la base de datos pues requieren de muchos recursos al momento de abrir o cerrar la conexión.

2. **Fábrica de Transacciones**.- Este servicio provee al usuario la habilidad de ejecutar más de una sentencia a la base de datos a la vez.

3. **Fábrica de Sesiones (Session Factory**).- La fábrica de sesiones es aquella que se encarga de decir al sistema donde se encuentran todos los ficheros de mapeo de NHibernate, el dialecto de NHibernate a utilizar varía según el gestor de base de datos, de este componente se generan los objetos Session de NHibernate, los cuales son los encargados de realizar todas las operaciones sobre la base de datos.

La persistencia de la información es la parte más crítica en una aplicación de software, si la aplicación está diseñada con orientación a objetos, la persistencia se logra por serialización del objeto o almacenando en una base de datos; las bases de datos más populares hoy en día son relacionales; el modelo de objetos difiere en muchos aspectos

del modelo relacional; la interface que une esos dos modelos se llama marco de mapeo relacional-objeto ORM.

La capa de persistencia es la encargada de abstraer y resolver el acceso a datos a un motor de base de datos relacional, su objetivo es ser la única que conoce como son persistidos los objetos de dominio de la aplicación y como recuperarlos abstrayendo el choque de impedancias entre objetos y tablas relacionales.

La capa superior no interactúa directamente con la base de datos, sino que lo hace mediante la interfaz expuesta por la capa de persistencia, logrando así la independencia buscada, esto permite cambiar la estrategia con que se persisten los objetos, incluso cambiar la tecnología o el motor utilizado sin impactar mas allá de ésta capa.

La capa de persistencia se expone a través de la interfaz, esta interfaz expone métodos como por ejemplo el crear un objeto de dominio, borrarlo, realizar búsquedas por clave y por distintos criterios; con ella se obtiene a una instancia concreta del broker de persistencia, éste broker puede ser visto como un ORM, que se obtiene a través de una Factory de ORMs que cumplen con dicha interfaz.

La persistencia de la información es la parte más crítica en una aplicación de software. Si la aplicación está diseñada con orientación a objetos, la persistencia se logra por: serialización del objeto o, almacenando en una base de datos.

Marcos de trabajo como Java o .Net han popularizado el uso de modelos de objetos UML en el diseño de aplicaciones dejando de lado el enfoque monolítico de una aplicación.

Las bases de datos más populares hoy en día son relacionales. Oracle, SQLServer, Mysql, Postgress son los DBMS más usados.

En el momento de persistir un objeto, normalmente, se abre una conexión a la base de datos, se crea una sentencia SQL parametrizada, se asignan los parámetros y recién allí se ejecuta la transacción.

Está demostrado que un 35% de tiempo de desarrollo de software está dedicado al mapeo entre objeto y su correspondiente relación, como se ve, la incongruencia entre los 2 modelos aumenta a medida que crece el modelo de objetos, hay varios puntos por considerar:

- ✓ Carga perezosa.
- ✓ Referencia Circular.
- ✓ Caché.
- ✓ Transacciones.

Se sabe que las tablas tienen atributos simples, o sea, tipo definidos previamente por los arquitectos del software, por otro lado, un objeto tiene tanto atributos simples como aquellos definidos por el usuario, que en sí es otro objeto más.

La incongruencia que se presenta entre el modelo relacional y el de objetos es la diferencia en la forma de representar atributos de los dos modelos, así en uno se tiene una representación tabular, mientras que en otro se tiene una representación jerárquica, a su vez la incongruencia que se da entre la tecnología de objetos y la relacional, fuerza al programador a mapear el esquema de objetos a un esquema de datos.

Para ello se usa una capa extra muy fina pero suficiente para servir como un puente entre los dos modelos para implementar esos mapeos, se necesita agregar código a los objetos de negocios, código que impacta en la aplicación.

Para la mayoría de las aplicaciones, almacenar y recuperar información implica alguna forma de interacción con una base de datos relacional, esto ha representado un problema fundamental para los desarrolladores porque algunas veces el diseño de datos relacionales y los ejemplares orientados a objetos comparten estructuras de relaciones muy diferentes dentro de sus respectivos entornos.

Esto permite que el código SQL este agrupado en estos files xml, permitiendo que sean fácilmente modificables y actualizables.

Algunos de los requerimientos buscados para la capa de persistencia son los siguientes:

- ✓ Manejar distintos tipos de mecanismos de persistencia
- ✓ Encapsular los mecanismos de persistencia utilizando métodos al estilo como por ejemplo save(obj), delete(obj), create(obj), retrieve(obj))
- ✓ Manejo de transacciones

- ✓ Identificación de objetos
- ✓ Utilización de Proxys
- ✓ Posibilidad de realizar consultas

3.8. PASOS DE INSTALACIÓN Y CONFIGURACIÓN DE NHIBERNATE

La instalación de NHibernate se lo puede realizar de dos maneras, de acuerdo a la manera como lo encontremos en internet, puede ser en un solo paquete de instalación o en archivo comprimido en donde se encuentra todas las librerías de NHibernate

PRIMERA FORMA DE INSTALACIÓN:

Empezaremos indicando la primera forma de instalación, en mi caso particular utilizaré el paquete de instalación de la versión de NHibernate 1.2.1.GA, que es la última versión completa liberada, junto con el instalador de la ayuda o documentación del mismo, tal y como se muestra en la Figura Nº III.13

Figura Nº III. 13. Paquete de Instalación de NHibernate 1.2.1G.A

Al ejecutar el paquete de instalación se nos presentará la pantalla presentada en la figura Nº III.14, indicándonos la versión de NHibernate

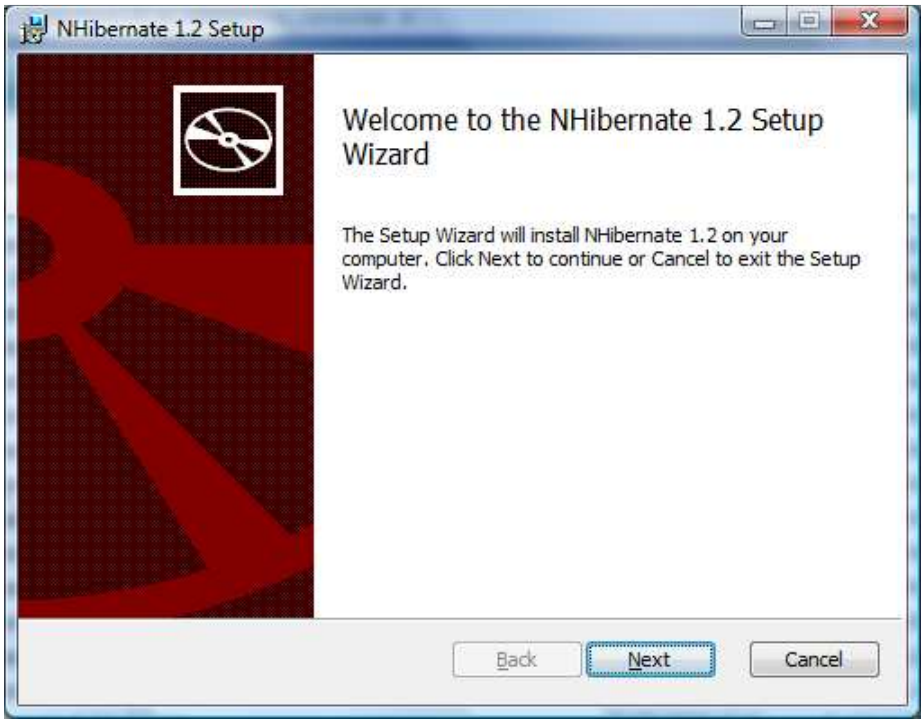


Figura Nº III. 14. Pantalla inicial de la instalación de NHibernate

En la figura Nº III.15, se presenta la siguiente pantalla de instalación en donde constan el acuerdo de licencias de instalación y uso de NHibernate

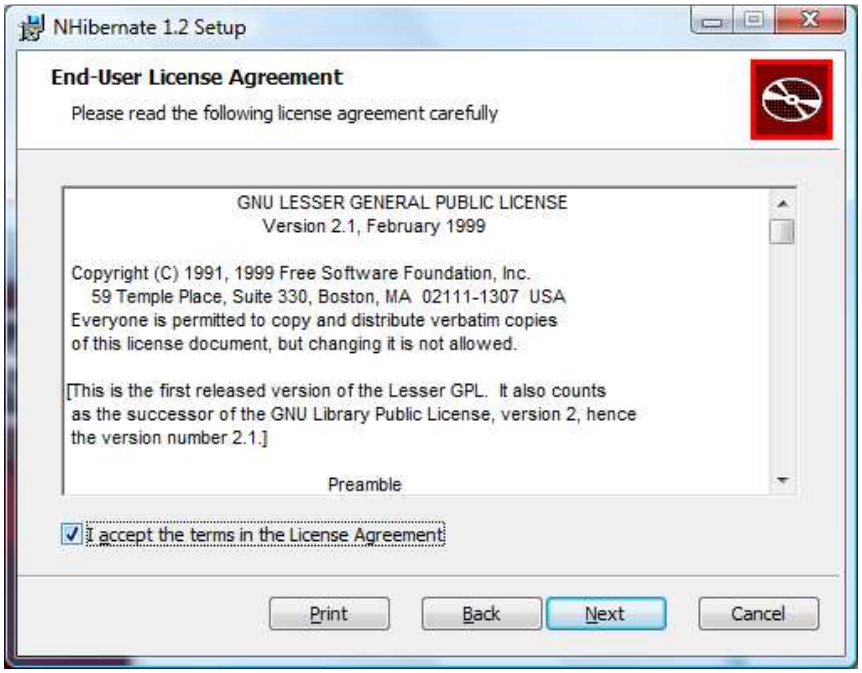


Figura Nº III. 15. Acuerdo de licencias de instalación de NHibernate

A continuación se presenta el árbol con las características para la instalación de NHibernate, en el que además se debe indicar la ubicación en donde se le va a instalar, tal y como se presenta en la figura Nº III. 16

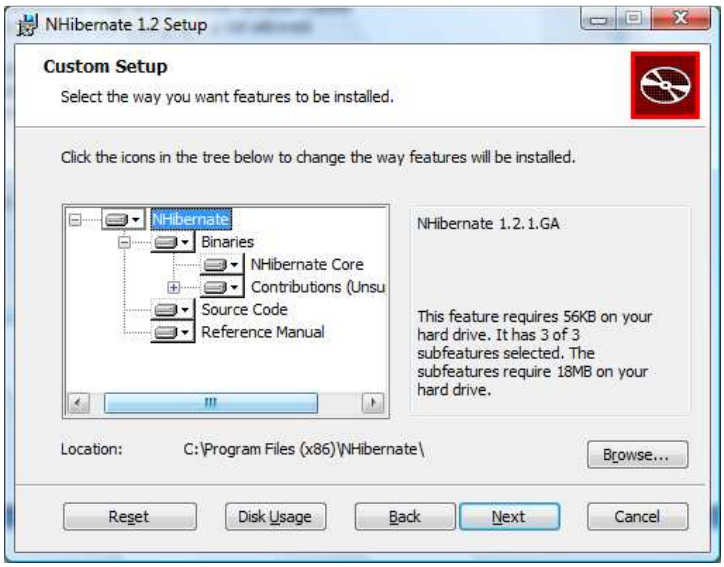


Figura Nº III. 16. Características de instalación de NHibernate

En la figura Nº III.17, se presenta la pantalla del proceso de instalación de NHibernate

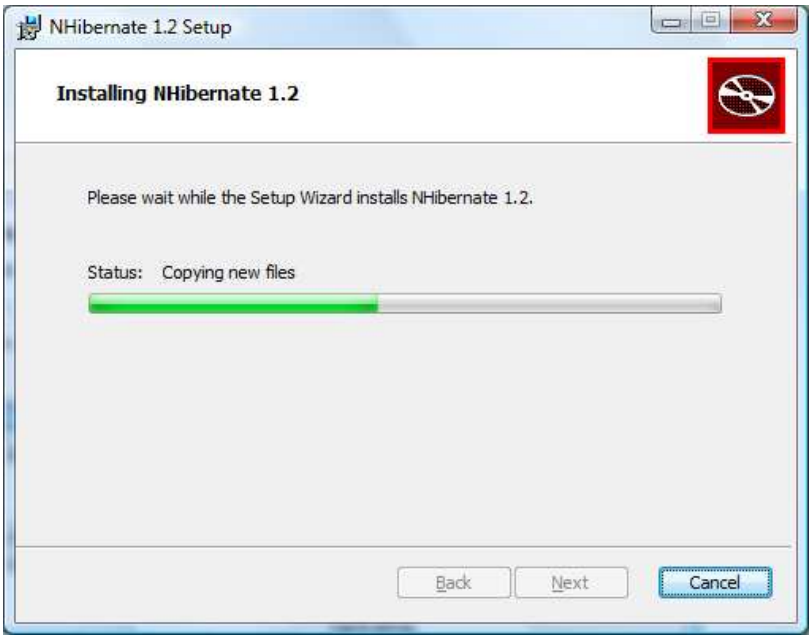


Figura Nº III. 17. Proceso de la instalación de NHibernate

Y finalmente se concluye la instalación con la pantalla plasmada en la figura Nº III.18

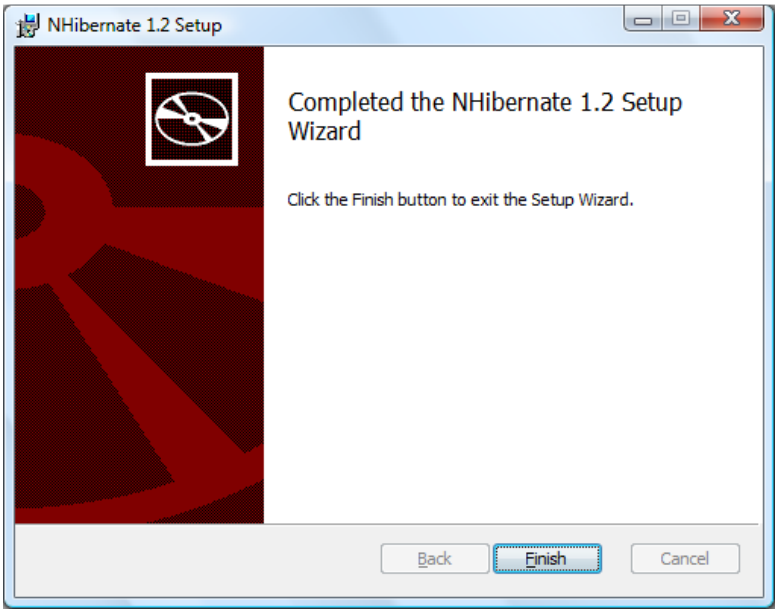


Figura Nº III. 18. Culminación de la instalación de NHibernate

Concluida la instalación del paquete de NHibernate, continuamos con la instalación de la ayuda o documentación del mismo, que se presenta en la figura Nº III.13, el inicio de la instalación empieza con la pantalla reflejada den la figura Nº III.19.

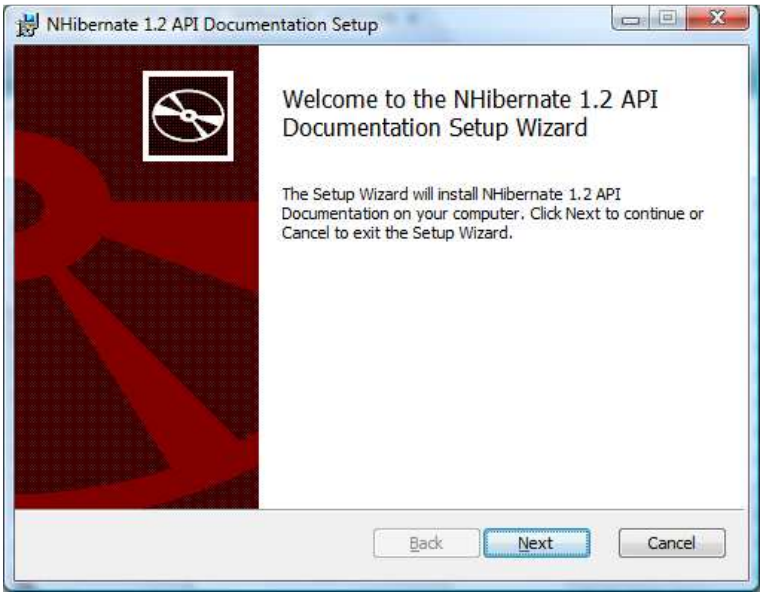


Figura Nº III. 19. Características de instalación de NHibernate

En la figura Nº III.20, se presentan los acuerdos de licenciamiento de NHibernate

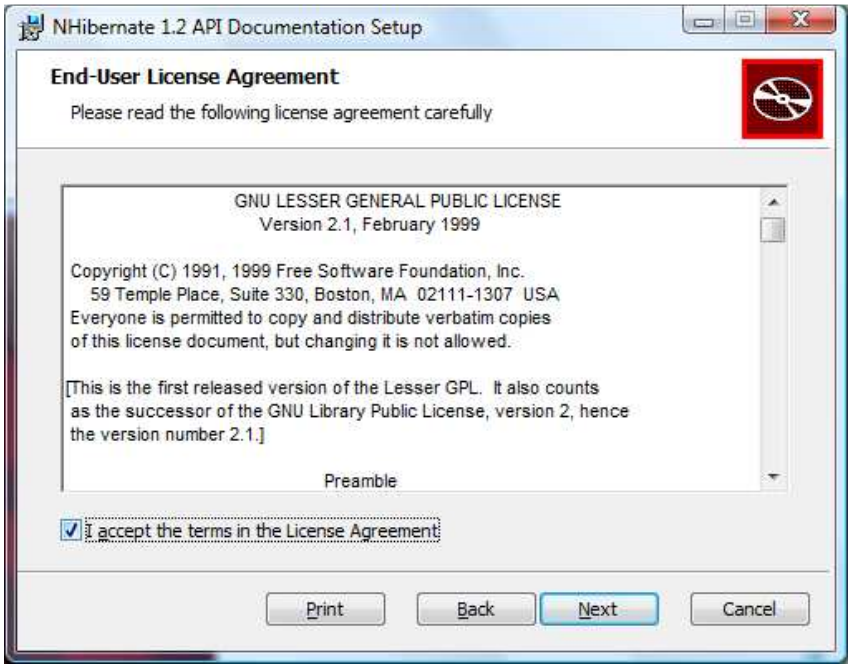


Figura Nº III. 20. Acuerdos de licencias de instalación de la Ayuda de NHibernate

A continuación se presenta la figura Nº III.21, con la pantalla de las características de NHibernate

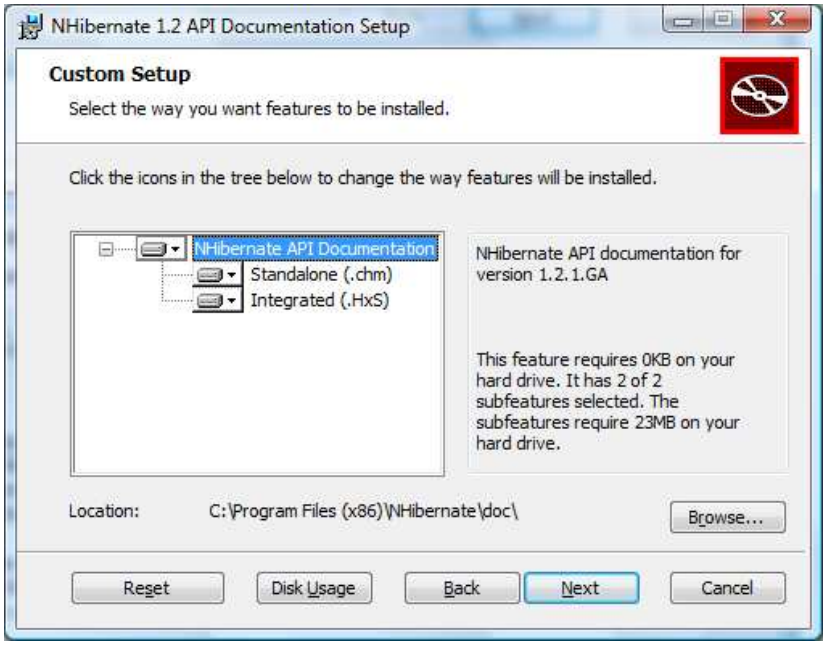


Figura Nº III. 21. Características de instalación de la ayuda de NHibernate

El proceso de instalación de la documentación de NHibernate 1.2 API se presenta en la figura Nº III.22

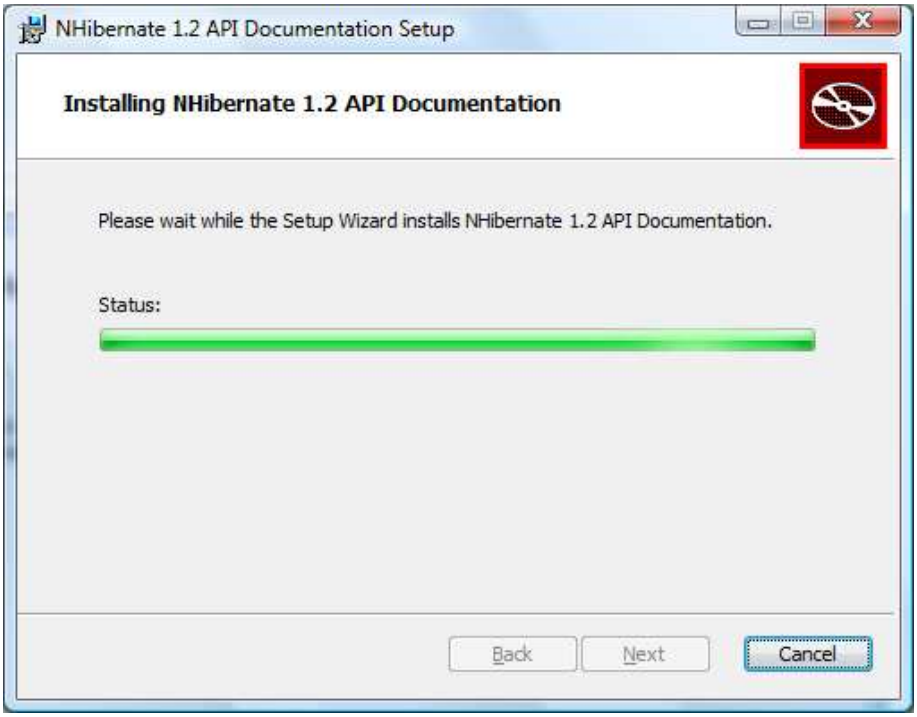


Figura Nº III. 22. Proceso de instalación de la ayuda de NHibernate

Siguiendo estos pasos la instalación lo podemos verificar en la ruta instalada como se muestra en la figura N III.23

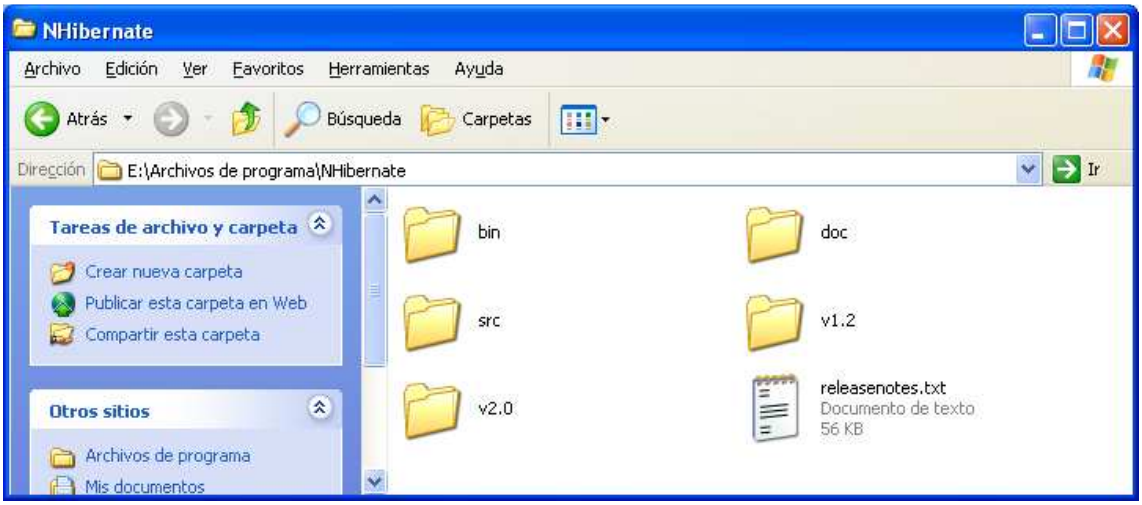


Figura Nº III. 23. Ruta de instalación de NHibernate

SEGUNDA FORMA DE INSTALACIÓN

Esta forma de instalación o uso de NHibernate es la más común, ya que es de esta manera como lo podemos encontrar en internet, en archivos comprimidos .zip, como se presenta en la figura Nº III.24

Figura Nº III. 24. Archivos comprimidos de las librerías de NHibernate

La carpeta NHibernate-1.2.1.GA-bin.zip, tiene las librerías necesarias con las que interactúa o trabaja NHibernate, por lo que es recomendable descomprimirle en la ubicación en donde se encuentra instalado Visual Studio .Net, que permitirá agregarlos a la solución del proyecto como referencia a las librerías necesarias, como se muestra en la figura Nº III.25, luego direccionamos en la ubicación en donde se encuentra instalado o ubicado la carpeta de NHibernate y agregamos a la solución como se presenta en la figura Nº III.26

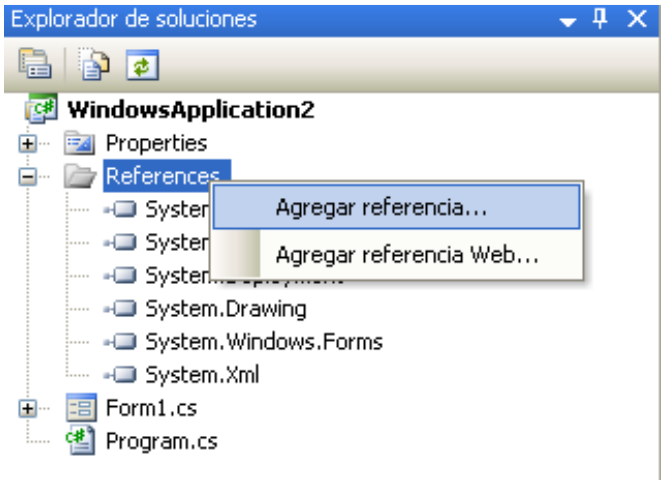


Figura Nº III. 25. Agregar referencia en la solución de .NET

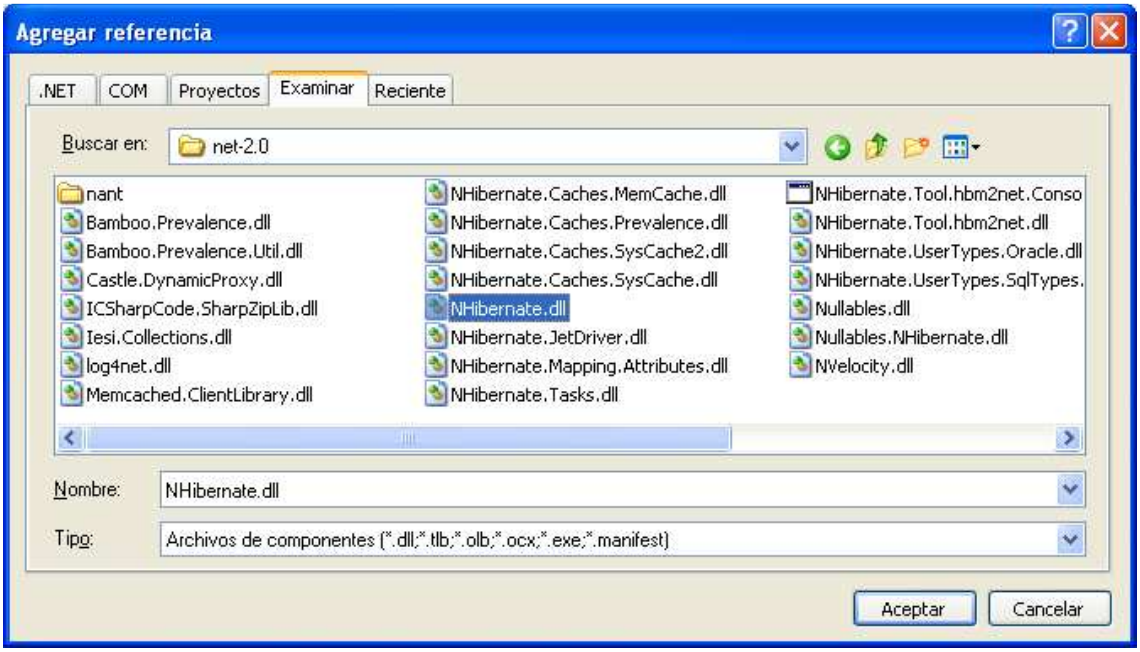


Figura Nº III. 26. Ubicación de las librerías de NHibernate

3.9. PATRÓN DE DISEÑO PARA .NET 2005

Los patrones de diseño proveen soluciones reusables a problemas de diseño conocidos, además no define código y no es específico a determinado dominio de programación sino que por el contrario brinda una solución genérica y testeada para una clase de

problemas de diseño similares. Asimismo, conllevan una terminología común que puede ser utilizada para documentar de forma sencilla y sistemática diseños propios.

Así como los patrones de diseño documentan soluciones a problemas de diseño comunes, los patrones para persistencia tienen un rol similar en el campo en cuestión, concretamente se presentan a continuación una serie de categorías para patrones de persistencia en base a lo siguiente:

Los Patrones de Separación describen estrategias para aislar los componentes de acceso a datos de otras partes de una aplicación. Este aislamiento brinda flexibilidad a la hora de seleccionar un modelo de datos de soporte, así como también cuando se realizan cambios en las distintas formas de acceso a datos para un sistema.

Los Patrones de Recursos por su parte describen estrategias para administrar los objetos involucrados en el acceso a bases relacionales. En concreto, pretenden solucionar temas de desempeño y semántica que frecuentemente surgen cuando se utiliza el acceso a través de una interfaz como ODBC o JDBC.

Los Patrones de Entrada/Salida describen soluciones que simplifican operaciones de entrada o salida de datos utilizando traducciones consistentes entre datos relacionales y los objetos de dominio relacionados.

Los Patrones de Cache presentan soluciones que reducen la frecuencia de las operaciones para acceso a datos almacenando datos comunes un una estructura de cache.

Los patrones en esta categoría usualmente se basan en cachear objetos de dominio en vez de datos físicos permitiendo así mayor eficiencia ya que la lógica de negocios trabaja con estos primeros directamente.

Finalmente, la última categoría de patrones, **Patrones de Concurrencia**, trata que sucede cuando múltiples usuarios realizan operaciones concurrentes que implican datos comunes.

A continuación aplicaremos el patrón de persistencia con cada una de sus categorías en base al desarrollo del sistema empleando el framework NHibernate.

Partiendo de la instalación de NHibernate, indicaremos paso a paso el proceso para conseguir el sistema terminado y funcionando perfectamente, para ellos detallamos las actividades que se realizarán en cada paso.

PATRÓN DE SEPARACIÓN

La estructura del proyecto constará de tres capas, como se ilustra en la figura Nº III.27

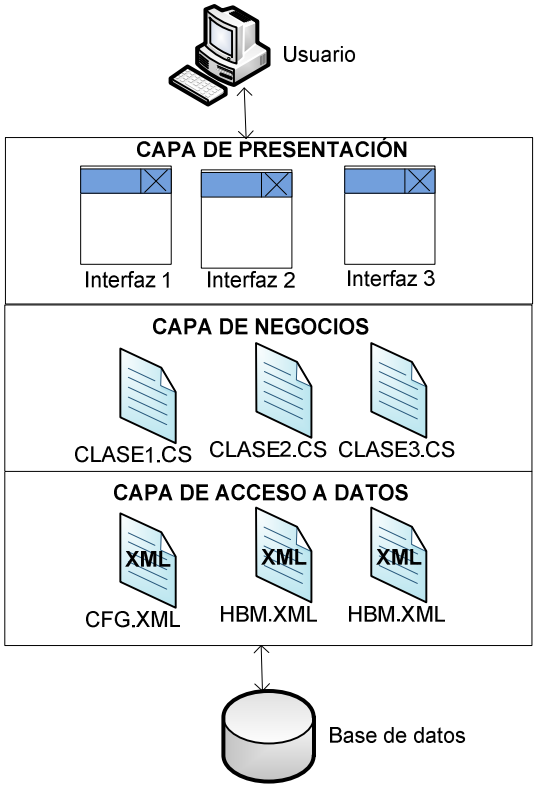


Figura Nº III. 27. Estructura del Proyecto

1. La capa inicial la de acceso a datos, la misma que contendrá los archivos de mapeo XML propios de NHibernate
2. La capa de negocios, en la que incluirá las entidades o clases del proyecto, las mismas servirán como un puente entre la capa de acceso a datos y la capa de presentación, ya que ésta posee las reglas de negocio, procesa los datos de la capa de presentación
3. Finalmente, la capa de de presentación es donde estarán los formularios del proyecto, que servirán para interactuar directamente con el usuario permitiendo el ingreso de los datos de usuario, muestra los datos, valida el ingreso de datos, etc

PATRÓN DE RECURSOS

Paso 1. Creación de un nuevo proyecto en Visual Studio .NET 2005, siguiendo el proceso como se indica en la figura Nº III.27

Figura Nº III. 28. Creación de una solución en Visual Studio .Net

Paso 2. Se nos presentará una pantalla como se indica en la figura Nº III.29, en donde se elige el tipo de proyecto, en mi caso particular seleccioné la opción de Visual C# para una aplicación Windows, en donde además se ingresa el nombre del proyecto para crear una aplicación con una interfaz de usuario de Windows, se seleccionará la ubicación física del proyecto y finalmente se ingresa el nombre de la solución, concluido esto se presiona el botón aceptar

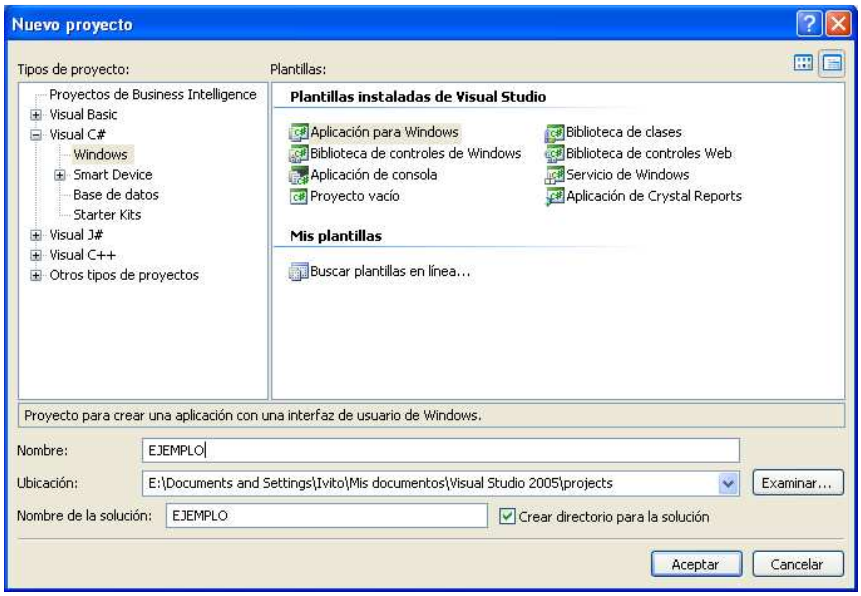


Figura Nº III. 29. Ingreso del nombre del proyecto en C#.net

Paso 3. Con el proyecto creado se procederá a agregar las referencias necesarias del framework NHibernate, como se indica en la figura Nº III.30, proceso que además se lo detallo en las figuras III.25 y III.26.

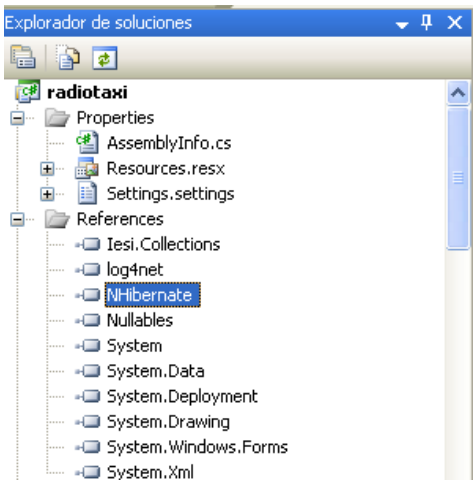


Figura Nº III. 30. Agregación de las referencias

Las librerías comunes que no deben faltar en un proyecto a desarrollarse con NHibernate son NHibernate, log4net y Iesi.Colelctions:

1. **NHibernate.dll.-** Se emplea para proyectos que implementen el framework Nhibernate.
2. **log4net.dll.-** Permite controlar un log de los eventos ocurridos durante la ejecución de una aplicación. Este log puede almacenarse en un archivo plano o en una base de datos.
3. **Iesi.Collections.dll:** Implementa algunas colecciones, como Sets, que no están disponibles en el .NET Framework.

Paso 4. En nuestro motor de base de datos, como SQL Server 2005 el que yo estoy empleando, creamos las tablas y el diagrama relacional, como se muestra en la figura en donde consta de siete tablas, tal y como se muestra en la figura Nº III.31

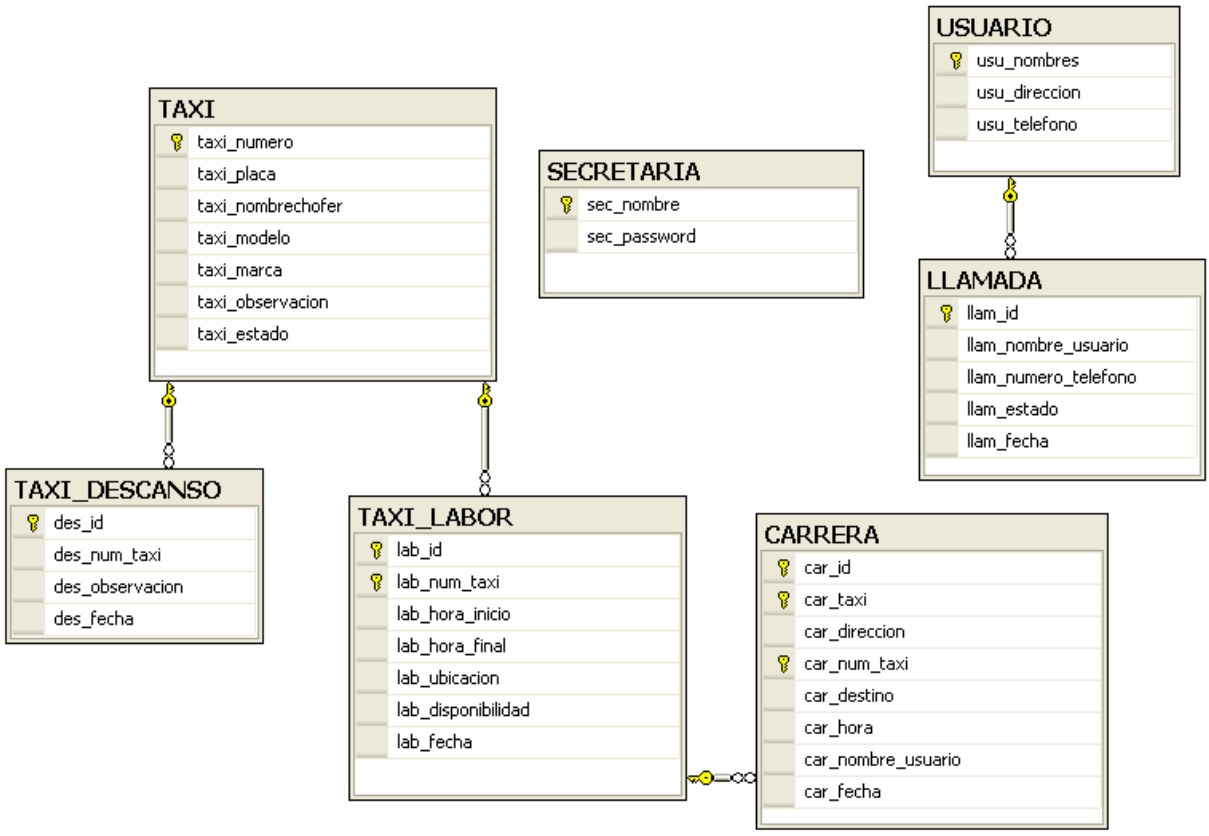


Figura Nº III. 31. Creación de las tablas y relaciones de la base de datos

PATRÓN DE ENTRADA/SALIDA

Para la capa de negocios existen dos patrones; los modelos de dominios y los script de transacciones que se utilizan cuando no se tienen objetos en la capa de negocios; presentado el caso se empleará el Modelo d Dominios ya que se trabajará con asociaciones y relaciones objetos y clase de negocio

Paso 5. Creamos el modelo relacional con las clases, teniendo como guía o referencia el modelo relacional de la base de datos, en donde primero debemos crear una carpeta que va a contener las clases o entidades del proyecto, luego agregamos las clases necesarias, dando clic derecho sobre la carpeta se desplegará un menú, se seleccionará Agregara y Nuevo Elemento, como se ilustra en la figura Nº III.32

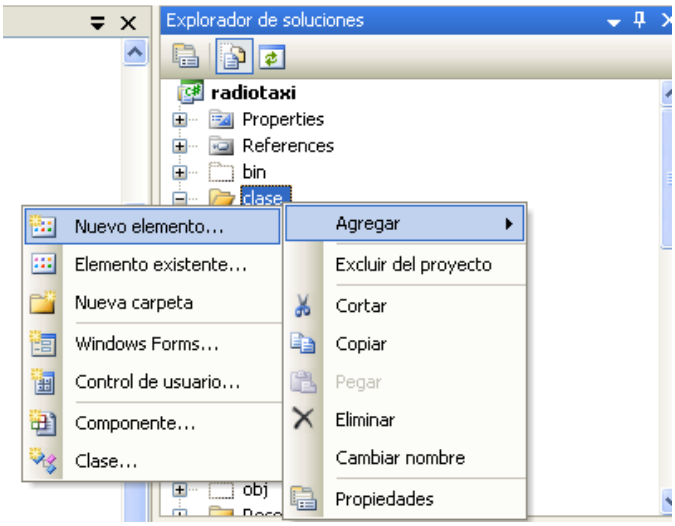


Figura Nº III. 32. Opciones para crear un Diagrama de clases

Paso 6. De las opciones se seleccionará Diagrama de clase en el que además se le ubicará su nombre, como se muestra en la figura Nº III.33

Figura Nº III. 33. Creación de un diagrama de clases

Paso 7. Se presentará la pantalla de la figura Nº III.34, en donde se empezará a crear una nueva clase, dando clic derecho sobre la pantalla y del menú se seleccionará Agregar y se tomará la opción clase

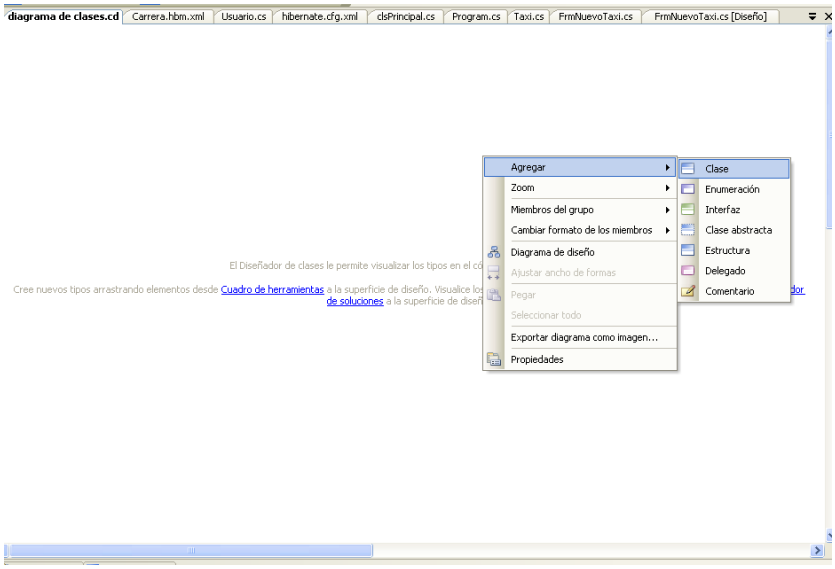


Figura Nº III. 34. Diagrama de clases - Opciones para la creación de clase

Paso 8. Siguiendo el paso 7, se nos presentará la pantalla de la figura Nº III.35, en donde se le ubicará el nombre de la clase, el tipo de la clase, ingresado estos datos se presionará el botón Aceptar

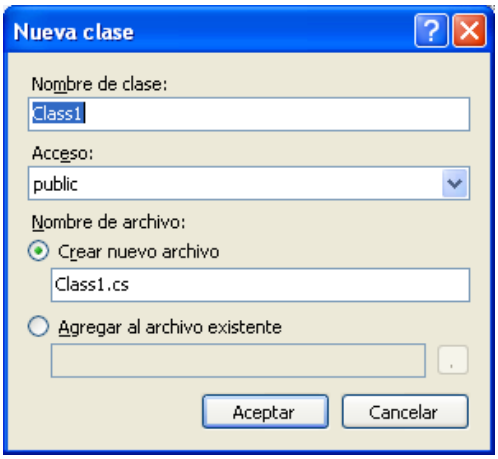


Figura Nº III. 35. Pantalla de Nueva Clase

Paso 9. En la clase creada según lo indicado en el paso anterior, empezamos a crear los campos, propiedades y métodos de la misma, como se presenta en la figura Nº III.36, se da clic derecho sobre la clase, del menú desplegado se seleccionará Agregar y de allí las opciones método, propiedad, campo, evento, constructor, destructor, constante dependiendo el caso.

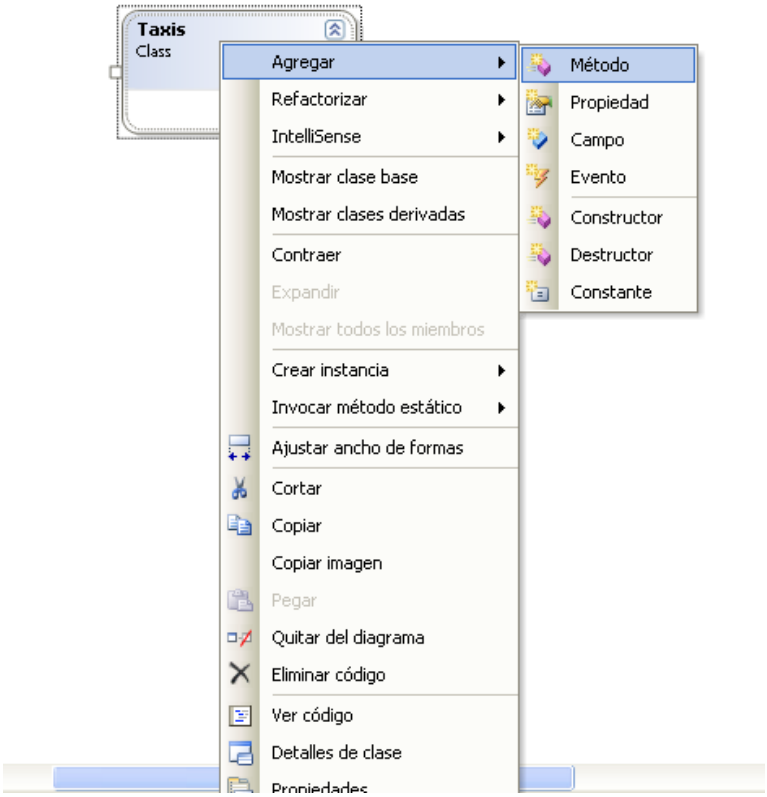


Figura Nº III. 36. Creación de Métodos y propiedades de la clase

Hasta conseguir obtener la estructura de la clase semejante a la tabla del modelo relacional de la base como se muestra en la figura Nº III.37

Figura Nº III. 37. Estructura de la clase

Paso 10. Los campos y propiedades de la clase creada se le podrá modificar los tipos de datos o a su vez alguna información con respecto a los mismos, en la opción de Detalle de la Clase como se muestra en la figura Nº III.38

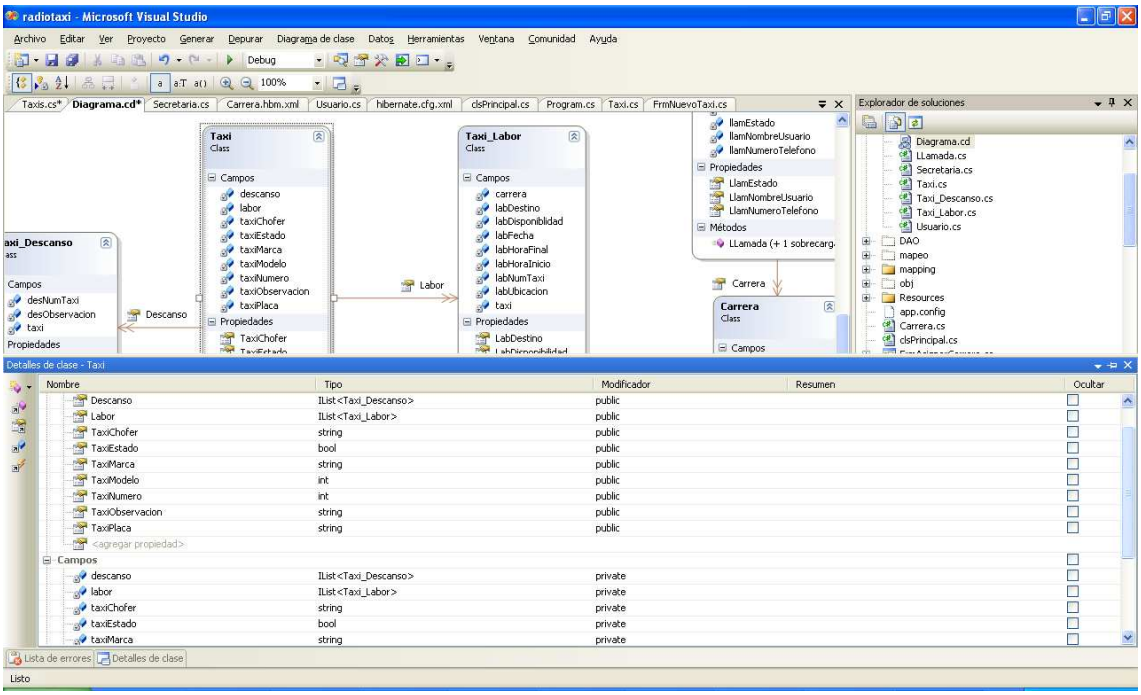


Figura Nº III. 38. Detalles de la estructura de la clase

Para conocer la estructura del código de la clase, se le dará doble sobre el nombre de la clase y automáticamente se accederá al mismo, en donde parte del código se presenta en la figura Nº III.39.

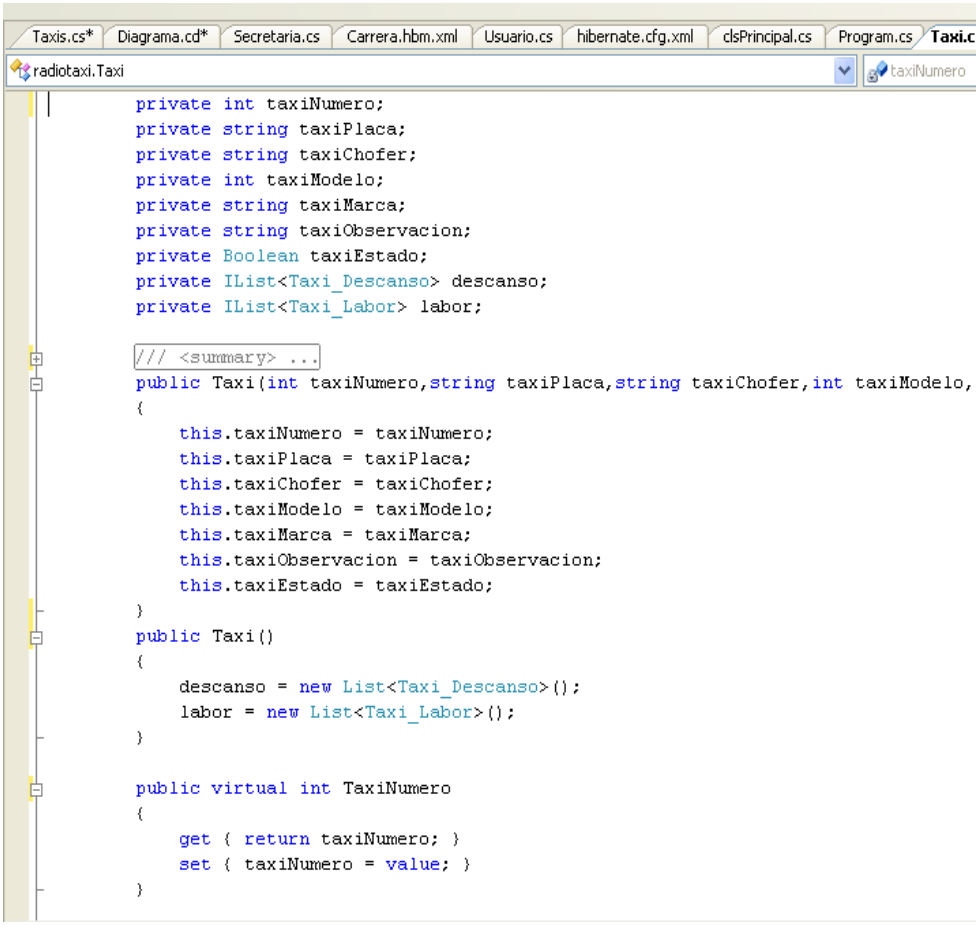


Figura Nº III. 39. Código de la clase creada

EL código de la clase TAXI presentado en la figura Nº III.39, es el siguiente:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace radiotaxi
{
    [Serializable]
    public class Taxi:IEquatable<Taxi>
    {
        private IList<Taxi_Descanso> descanso;
        private IList<Taxi_Labor> labor;
        private string taxiChofer;
        private Boolean taxiEstado;
        private string taxiMarca;
        private int taxiModelo;
        private int taxiNumero;
        private string taxiObservacion;
        private string taxiPlaca;
```

```
 public Taxi(int taxiNumero, string taxiPlaca, string taxiChofer, int
taxiModelo, string taxiMarca, string taxiObservacion, Boolean
taxiEstado): this()
        {
            this.taxiNumero = taxiNumero;
            this.taxiPlaca = taxiPlaca;
            this.taxiChofer = taxiChofer;
            this.taxiModelo = taxiModelo;
            this.taxiMarca = taxiMarca;
            this.taxiObservacion = taxiObservacion;
            this.taxiEstado = taxiEstado;
        }
        public Taxi()
        {
            descanso = new List<Taxi_Descanso>();
            labor = new List<Taxi_Labor>();
        }

        public virtual int TaxiNumero
        {
            get { return taxiNumero; }
            set { taxiNumero = value; }
        }

        public virtual string TaxiPlaca
        {
            get { return taxiPlaca; }
            set { taxiPlaca = value; }
        }

        public virtual string TaxiChofer
        {
            get { return taxiChofer; }
            set { taxiChofer = value; }
        }

        public virtual int TaxiModelo
        {
            get { return taxiModelo; }
            set { taxiModelo = value; }
        }

        public virtual string TaxiMarca
        {
            get { return taxiMarca; }
            set { taxiMarca = value; }
        }

        public virtual string TaxiObservacion
        {
            get { return taxiObservacion; }
            set { taxiObservacion = value; }
        }

        public virtual bool TaxiEstado
        {
            get { return taxiEstado; }
            set { taxiEstado = value; }
        }
```

```
        public virtual IList<Taxi_Labor> Labor
        {
            get { return labor; }
            set { labor = value; }
        }
        public virtual IList<Taxi_Descanso> Descanso
        {
            get { return descanso; }
            set { descanso = value; }
        }
        public override string ToString()
        {
            return this.TaxiNumero.ToString();
        }
        public override int GetHashCode()
        {
            return taxiNumero.GetHashCode();
        }
        public virtual bool Equals(Taxi other)
        {
            if (this == other) return true;
            if (this.taxiNumero == other.taxiNumero) return true;
            return false;
        }
    }
}
```

Paso 10. Con los pasos seguidos anteriormente se logra obtener el árbol de las clases y diagramas de clase como la presentada en la figura Nº III.40

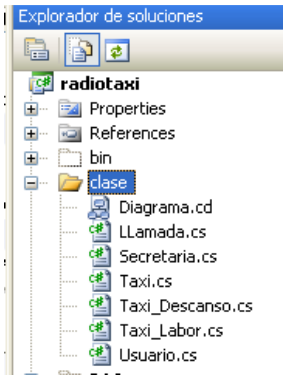


Figura Nº III. 40. Listado de las clases creadas

La estructura del diagrama de clases será como la presentada en la figura Nº III.41, con sus respectivas clases, campos, propiedades, métodos, relaciones

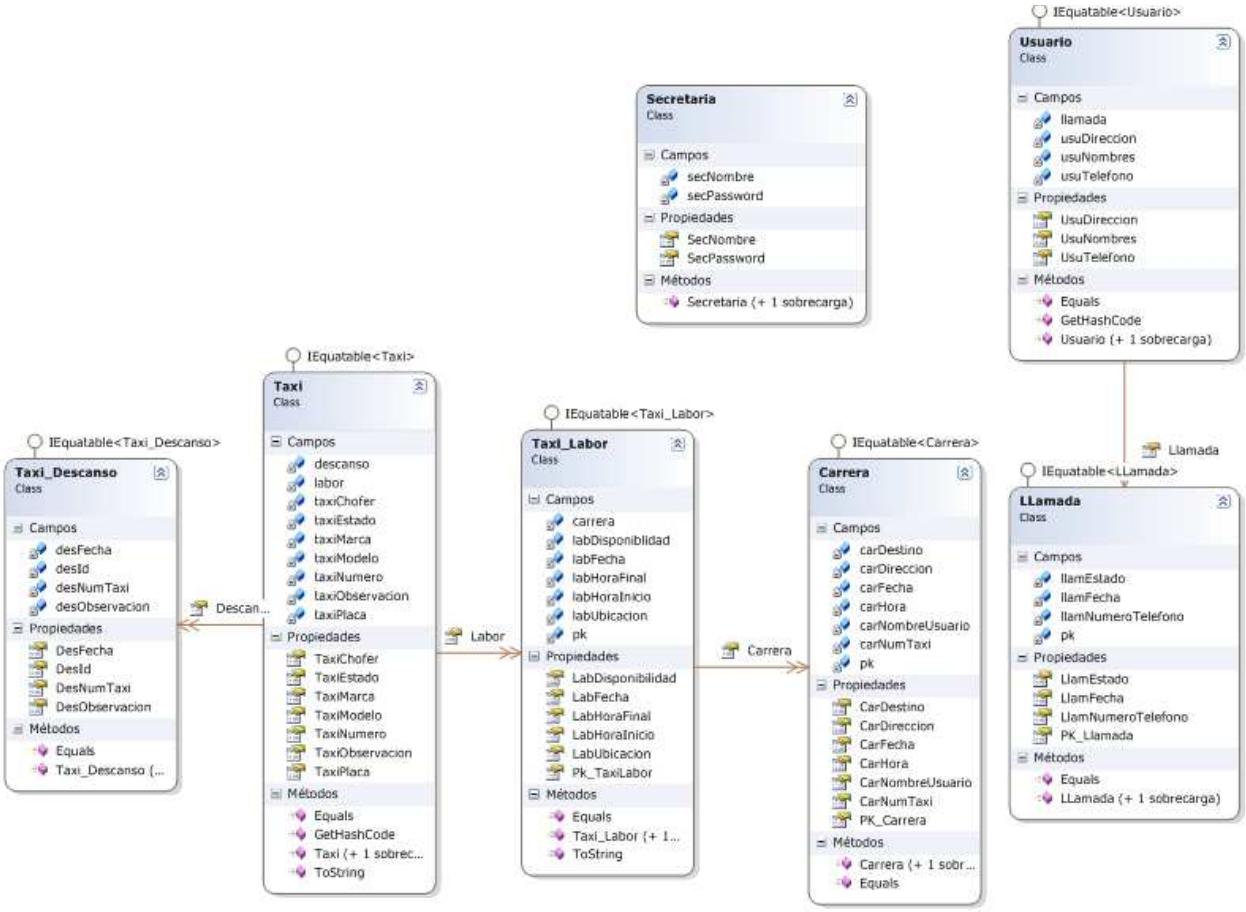


Figura Nº III. 41. Diagrama de clases en Visual Studio .NET 2005

PATRÓN DE CACHE

Para la capa de acceso a datos existen dos tipos de patrones: Los registros activos que constan los métodos de persistencia en la misma clase de negocio; y el Mapeo de datos que consta de que los objetos de negocio mantengan el código tradicional o POCO, y las clases con las operaciones CRUD independiente de ella; dada esta situación se empleará el patrón de Mapeo de Datos.

Paso 11. Ahora se creará los archivos de mapeo que necesita NHibernate para interpretar e interactuar con las tablas de la base de datos, para aquello se creará una carpeta que se llamará mapeo, en ella se dará clic derecho, del menú se elegirá Agregar y de ella Nuevo Elemento, tal y como se muestra en la Figura Nº III.42

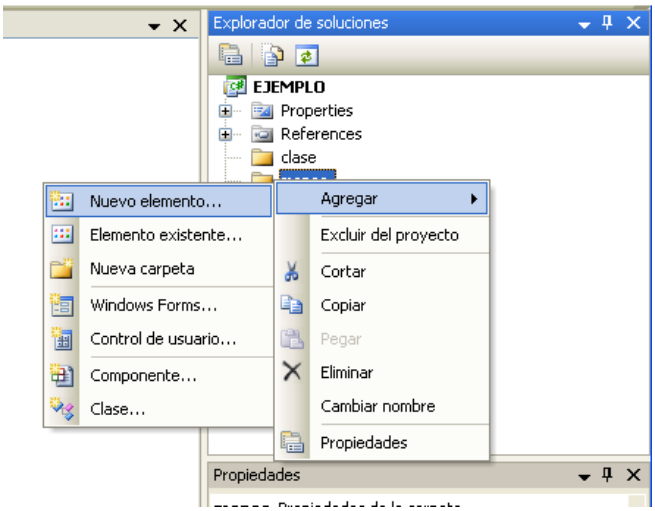


Figura Nº III. 42. Opciones para crear el archivo de mapeo XML

Paso 12. Siguiendo el paso anterior, se nos presentará una pantalla como la que se indica en la figura Nº III.43, de la misma seleccionamos Archivo XML y le ingresamos un nombre con la extensión HBM.XML, que nos indica que es un archivo de mapeo de NHibernate.

Figura Nº III. 43. Creación del archivo XML

Paso 12. En la figura Nº III.44, se muestra el listado de todos los archivos de mapeo creados dentro de la carpeta mapeo

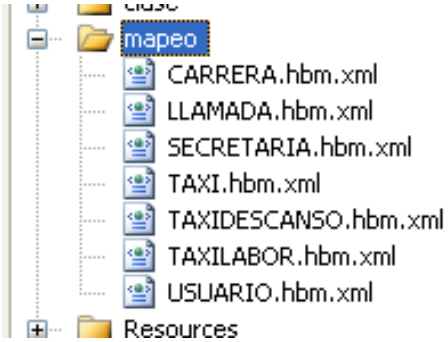


Figura Nº III. 44. Listado de los archivos de mapeo HBM.XML

Paso 13. De acuerdo a la estructura de la tabla en el modelo relacional, estructuramos el archivo de mapeo indicando primeramente el tipo de mapeo, a la clase de negocio con la que se va a vincular, el nombre del id de la clase con el tipo de dato a la que pertenece, y demás atributos de la misma, como se presenta en la figura Nº III.45.

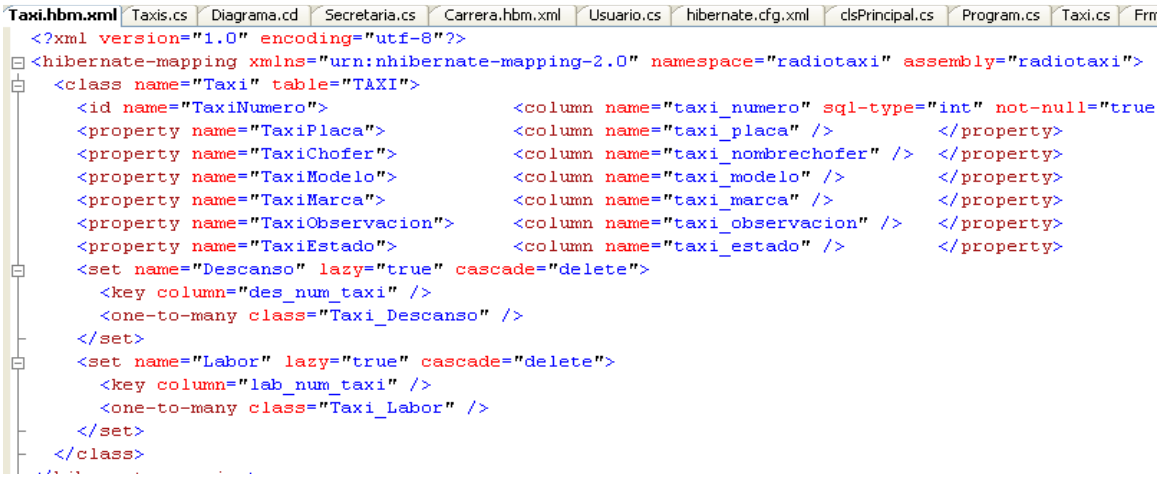


Figura Nº III. 45. Código del archivo de mapeo

El código del archivo de mapeo de Taxi.hbm.xml, mostrado en la figura Nº III.45, se presenta a continuación.

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.0" namespace="radiotaxi"
assembly="radiotaxi">
  <class name="Taxi" table="TAXI">
    <id name="TaxiNumero">                <column name="taxi_numero" sql-
type="int" not-null="true" length="4" />   <generator class="uuid.hex" />
</id>
    <property name="TaxiPlaca">           <column name="taxi_placa" />
</property>
    <property name="TaxiChofer">          <column name="taxi_nombrechofer" />
</property>
    <property name="TaxiModelo">          <column name="taxi_modelo" />
</property>
    <property name="TaxiMarca">           <column name="taxi_marca" />
</property>
    <property name="TaxiObservacion">     <column name="taxi_observacion" />
</property>
    <property name="TaxiEstado">          <column name="taxi_estado" />
</property>
    <set name="Descanso" lazy="true" cascade="delete">
      <key column="des_num_taxi" />
      <one-to-many class="Taxi_Descanso" />
    </set>
    <set name="Labor" lazy="true" cascade="delete">
      <key column="lab_num_taxi" />
      <one-to-many class="Taxi_Labor" />
    </set>
  </class>
</hibernate-mapping>
```

Paso 14. En todos los archivos de mapeo, se le debe cambiar las propiedades, en especial la acción de generación a recurso incrustado, esto para que el fichero xml se

incruste en el assembly a la hora de compilar, al igual que la propiedad copiar en directorio de resultados le cambiamos a “copiar siempre”, como se indica en la figura Nº III.46.

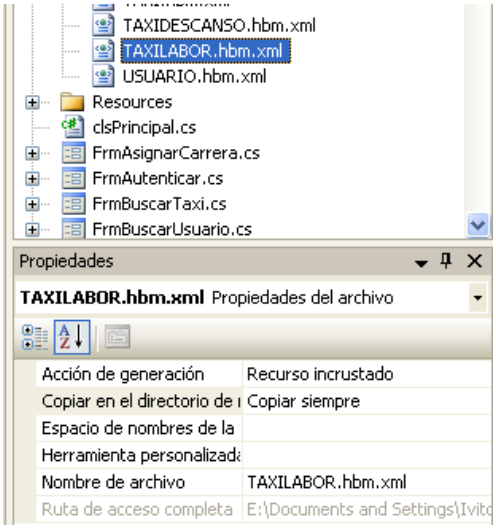


Figura Nº III. 46. Cambio de la propiedad del archivo de mapeo

Paso 15. Se creará el archivo de configuración del NHibernate, aunque existe diversas formas de conexión, pero se explicará la más simple y más utilizada, este archivo toma de nombre hibernate.cfg.xml para aplicaciones windows, ó app.config.xml para aplicaciones web, en donde enlaza a todas las capas del proyecto, el mismo que posee la cadena de conexión con la base de datos y con el ensamblador del proyecto, para aquello del nombre de la solución se da clic derecho, de las opciones se selecciona Agregar y dentro de ella Nuevo Elemento

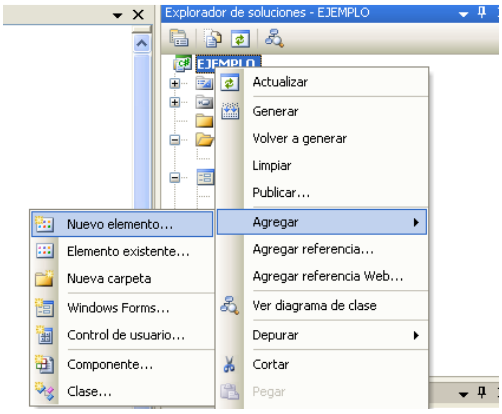


Figura Nº III. 47. Opciones para la creación del archivo de configuración

Paso 16. Siguiendo el paso anterior se nos presentará la siguiente pantalla ilustrada en la figura Nº III.48, en la que se elige Archivo XML y además se le pondrá el nombre de hibernate.cfg.xml, ya que este será el archivo de configuración, finalmente se presionará el botón agregar.

Figura Nº III. 48. Creación del archivo de configuración

Paso 17. En el archivo creado ingresamos en código mostrado en la figura Nº III.49, el cual será configurado de acuerdo a las exigencias del proyecto

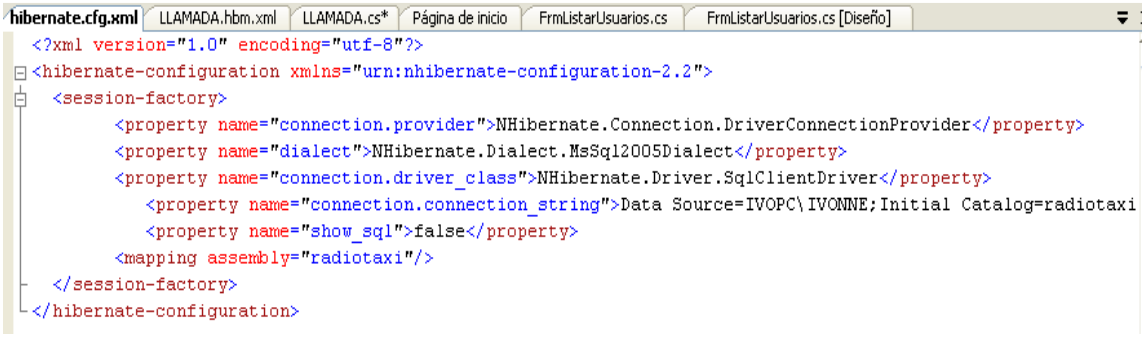


Figura Nº III. 49. Código del archivo de configuración NHibernate.cfg.xml

El código del archivo de configuración de NHibernate presentado en la figura Nº III.49, se muestra a continuación

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
  <session-factory>
     <property name="connection.provider">
          NHibernate.Connection.DriverConnectionProvider
     </property>
     <property name="dialect"> NHibernate.Dialect.MsSql2005Dialect</property>
    <property name="connection.driver_class">NHibernate.Driver.SqlClientDriver</property>
     <property name="connection.connection_string">
          Data Source=IVOPC\IVONNE;Initial Catalog=radiotaxi;Integrated Security=True
     </property>
     <property name="show_sql">false</property>
     <mapping assembly="radiotaxi"/>
  </session-factory>
</hibernate-configuration>
```

Para realizar este archivo de configuración se deberá conocer las características o propiedades que las compone, las mismas que las detallamos:

CONNECTION.PROVIDER.- Es el proveedor de la conexión, en éste caso un driver

DIALECT.- Dialecto SQL del sistema de base de datos: MsSql 2005, el cual le permite que genere un SQL optimizado para una base de datos relacional en particular.

CONNECTION.DRIVER_CLASS.- Clase que contiene el driver SqlClient para SQL Server.

CONNECTION.CONNECTION_STRING.- Cadena de conexión a utilizar para obtener una conexión con una base de datos

SHOW_SQL.- Habilita el registro de todas las instrucciones SQL generadas en la consola

Paso 18.- En el archivo de configuración, se deberá cambiar sus propiedades, para lo cual nos ubicamos en el archivo, se da clic derecho y elegimos propiedades, en donde en la Acción de generación le cambiamos a "contenido" para que en el momento de la compilación tome en consideración el archivo de configuración

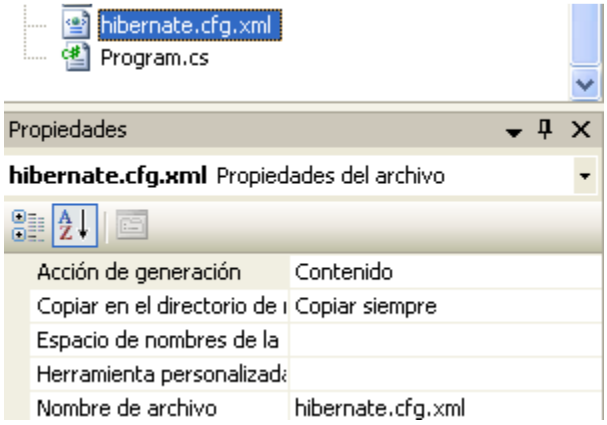


Figura Nº III. 50. Cambio de la propiedad del archivo de configuración

Ahora ya se tiene la configuración del archivo de NHibernate, entonces se empezará a utilizar estos archivos según lo requerido.

Paso 19.- Se realizara la configuración con la utilización de los archivos de configuración y las sesiones para la persistencia de los objetos, en la figura Nº III.51, se

presenta algo de código de éste proceso en donde en una clase principal va a contener la configuración de las sesiones que permiten interactuar con el fichero hibernate.cfg.xml y con todos los ficheros xml que se requiera.

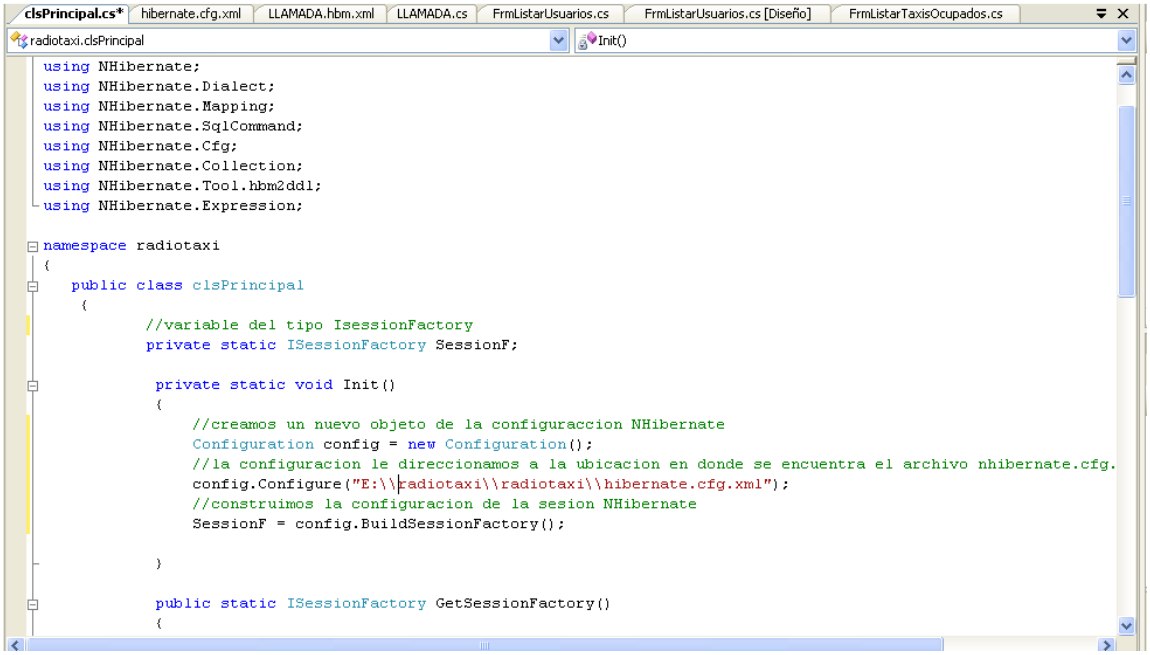


Figura Nº III. 51. Direccionamiento del archivo de configuración

El código presentado en la figura Nº III.51, se lo podrá apreciar a continuación:

```
using System;
using System.Collections.Generic;
using System.Text;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;


namespace radiotaxi
{
  public class clsPrincipal
   {
        private static ISessionFactory SessionF;

         private static void Init()
         {
            Configuration config = new Configuration();
            config.Configure("E:\\Documents and Settings\\Ivito\\Escritorio\\APLICACION para
modif\\radiotaxi\\radiotaxi\\hibernate.cfg.xml");
            SessionF = config.BuildSessionFactory();
```

```
        }

//creacion de sesiones
        public static ISessionFactory GetSessionFactory()
        {
            if (SessionF == null)
                Init();
            return SessionF;
        }


        public static ISession GetNewSession()
        {
            return GetSessionFactory().OpenSession();
        }
    }
}
```

Desarrollo de la Aplicación con NHibernate

Paso 20.- Se empezará a desarrollar la aplicación empleando lo antes realizado, se diseñará las interfaces para la capa de presentación, para lo cual se presiona clic derecho sobre el nombre de la solución, se selecciona la opción Agregar y Windows Form, como se presenta en la figura Nº III.52.

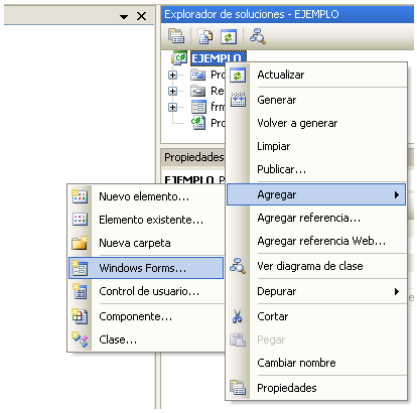


Figura Nº III. 52. Opciones para agregar WinForm

Paso 21.- La forma o interfaz de Windows inicial se presentará de la forma como se muestra en la figura Nº III.53, a la que se le incluirá las herramientas necesarias para el formulario

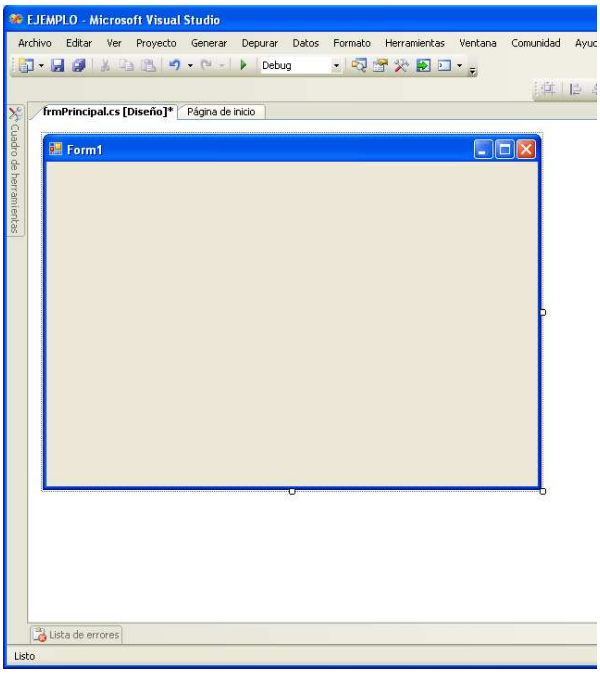


Figura Nº III. 53. Creación de WinForm al proyecto

Paso 22.- Se diseño la interfaz para insertar un usuario, como se muestra en la figura Nº III.54, la misma que esta basada en los campos que se posee en la clase Usuario.

Figura Nº III. 54. Interfaz para la inserción de datos

Paso 23.- Una vez que se haya diseñado la interfaz, se empezará a codificar en ella para conseguir el ingreso de los datos a través de la interfaz a la base de datos, para conseguir éste propósito se agregará las librerías propias de NHibernate, que servirán

para trabajar con los sesiones, con el archivo de configuración y con los archivos de mapeo, como se indica en la figura Nº III.55.

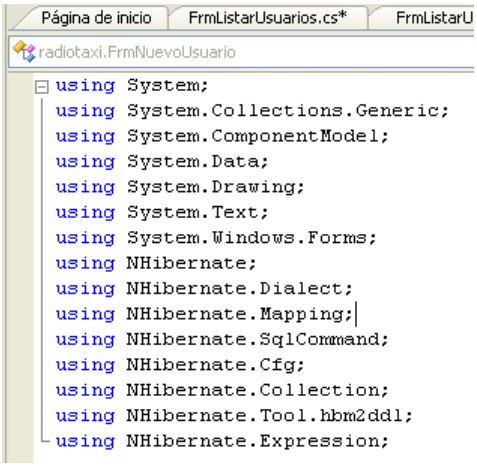


Figura Nº III. 55. Agregación de las librerías en el proyecto

Paso 24.- En el código del formulario, se realizará el inicio de las sesiones, como se indica en la figura Nº III.56

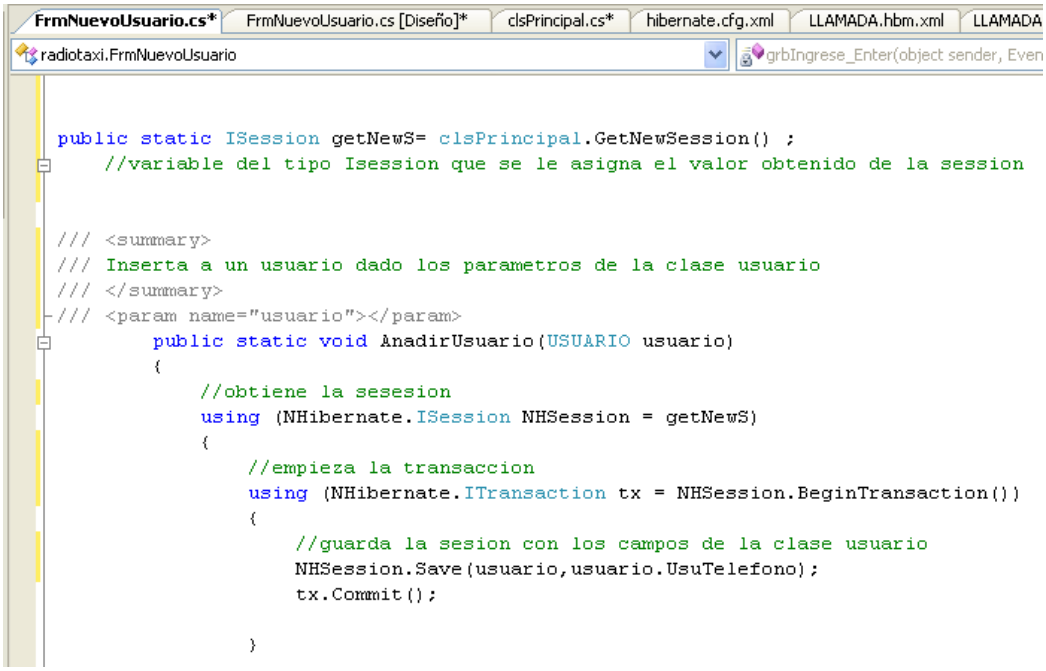


Figura Nº III. 56. Creación del método para la creación de usuarios

Paso 25.- Además se incluirá el proceso para generar el método insertar, representado en la Figura Nº III.57.

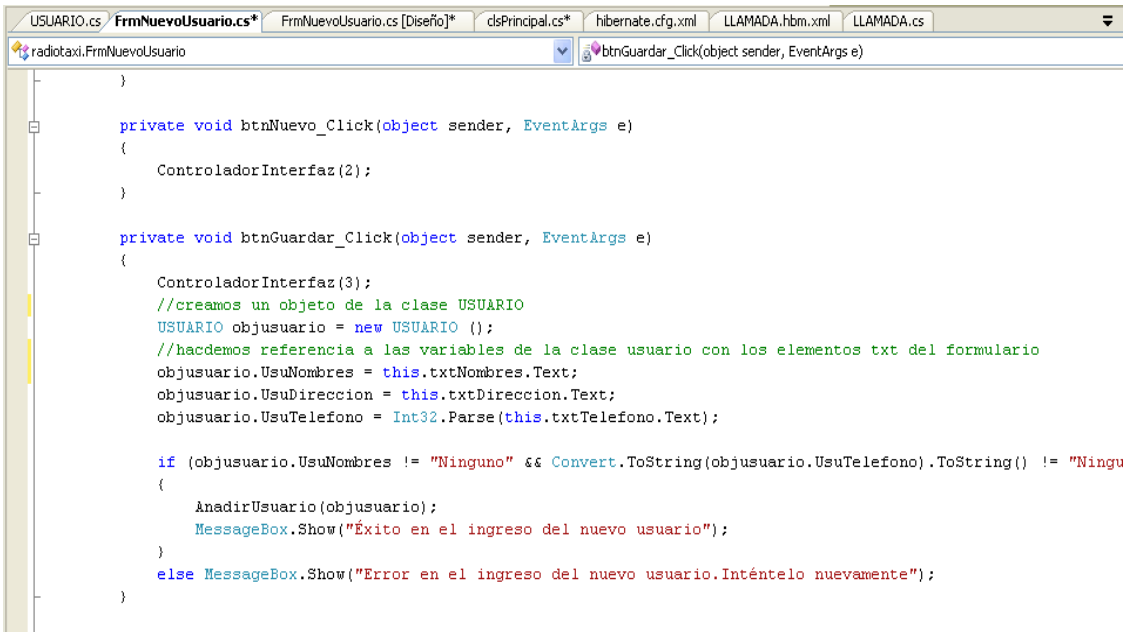


Figura Nº III. 57. Asignación de los elementos para ejecutar el método insertar

El código que se encuentra en el formulario para insertar un usuario representado en las IIII.57, se muestra a continuación:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;


namespace radiotaxi
{
   public partial class FrmNuevoUsuario : Form
   {
      public FrmNuevoUsuario()
      {
```

```
            InitializeComponent();
        }


/// <summary>
/// Inserta a un usuario dado los parametros de la clase usuario
/// </summary>
/// <param name="usuario"></param>
        public static void AnadirUsuario(Usuario usuario)
        {

            //variable del tipo Isession que se le asigna el valor obtenido de la session
            ISession getNewS= clsPrincipal.GetNewSession() ;

            //obtiene la sesesion
           using (NHibernate.ISession NHSession = getNewS )
           {
              //empieza la transaccion
              using (NHibernate.ITransaction tx = NHSession.BeginTransaction())
              {
                 //guarda la sesion con los campos de la clase usuario
                 NHSession.Save(usuario,usuario.UsuNombres);
                 tx.Commit();
              }
        }

        private void txtNombres_KeyPress(object sender, KeyPressEventArgs e)
        {
           if (Char.IsDigit(e.KeyChar) == true)//detecta si se ingresa letras
           {
              MessageBox.Show("No puede ingresar números", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
              e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
           }
        }

        private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
        {
           if (Char.IsLetter(e.KeyChar) == true)//detecta si se ingresa letras
           {
              MessageBox.Show("No puede ingresar letras", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
              e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
           }
        }

        private void txtNombres_Validating(object sender, CancelEventArgs e)
        {
           if ((this.txtNombres.Text == "Ninguno") || (this.txtNombres.Text == ""))//si no se ha ingresado datos
           { //MessageBox.Show("Ingrese la direccion");
              e.Cancel = true;//cancelece Para q pase al siguiente control
             //muestra un mensaje de error
              this.erpError.SetError(this.txtNombres, "Ingrese los Nombres del Usuario");            }
           else
              this.erpError.SetError(this.txtNombres, "");
        }

        private void txtDireccion_Validating(object sender, CancelEventArgs e)
        {
           if ((this.txtDireccion.Text == "Ninguno") || (this.txtDireccion.Text == ""))//si no se ha ingresado datos
           { //MessageBox.Show("Ingrese la direccion");
              e.Cancel = true;//cancelece Para q pase al siguiente control
              //muestra un mensaje de error
              this.erpError.SetError(this.txtDireccion, "Ingrese la Dirección del Usuario");
```

```
            }
            else
                this.erpError.SetError(this.txtDireccion, "");
        }

        private void txtTelefono_Validating(object sender, CancelEventArgs e)
        {
            if ((this.txtTelefono.Text == "Ninguno") || (this.txtTelefono.Text == ""))//si no se ha ingresado datos
            { //MessageBox.Show("Ingrese la direccion");
                e.Cancel = true;//cancelece Para q pase al siguiente control

                this.erpError.SetError(this.txtTelefono, "Ingrese el Teléfono del Usuario");//muestra un mensaje
de error

            }
            else
                this.erpError.SetError(this.txtTelefono, "");
        }

        private void FrmNuevoUsuario_Load(object sender, EventArgs e)
        {
            ControladorInterfaz(1);
        }

        private void btnNuevo_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(2);
        }

        private void btnGuardar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(3);
            //creamos un objeto de la clase USUARIO
            Usuario objusuario = new Usuario ();
            //hacdemos referencia a las variables de la clase usuario con los elementos txt del formulario
            objusuario.UsuNombres = this.txtNombres.Text;
            objusuario.UsuDireccion = this.txtDireccion.Text;
            objusuario.UsuTelefono = Int32.Parse(this.txtTelefono.Text);

            if (objusuario.UsuNombres != "Ninguno" && Convert.ToString(objusuario.UsuTelefono).ToString()
!= "Ninguno" && objusuario.UsuDireccion != "Ninguno")
            {
                AnadirUsuario(objusuario);
                MessageBox.Show("Éxito en el ingreso del nuevo usuario");
            }
            else MessageBox.Show("Error en el ingreso del nuevo usuario.Inténtelo nuevamente");
        }
```

Paso 26.- El proceso para generar selección de una clase; o para realizar el proceso de listar todos los elementos de una tabla se presenta en la figura Nº III.58, en el que se encuentra incluida la asignación de los datos abstraídos al DataGrid del formulario Listar Usuarios

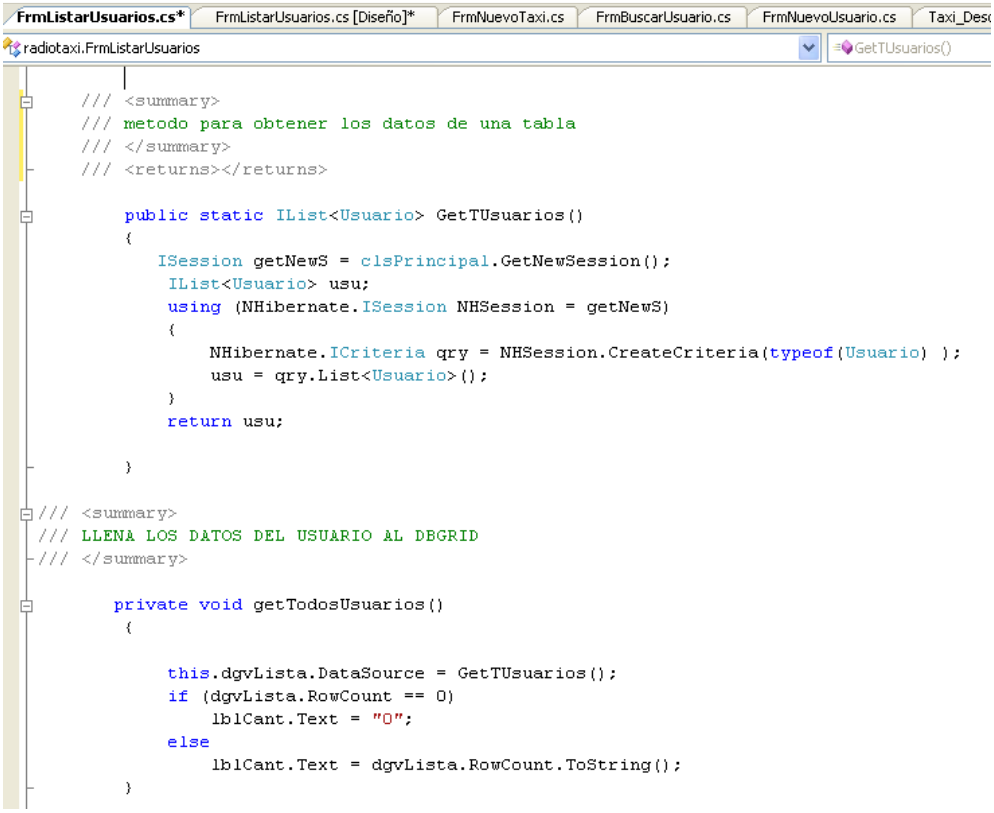


Figura Nº III. 58. Pantalla con el código para el método SELECT

Todo el código reflejado en la figura Nº III.58, se lo detalla a continuación

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;


namespace radiotaxi
{
   public partial class FrmListarUsuarios : Form
   {
      public FrmListarUsuarios()
      {
         InitializeComponent();
      }
```

```
    /// <summary>
    /// metodo para obtener los datos de una tabla
    /// </summary>
    /// <returns></returns>

        public static IList<Usuario> GetTUsuarios()
        {
           ISession getNewS = clsPrincipal.GetNewSession();
            IList<Usuario> usu;
            using (NHibernate.ISession NHSession = getNewS)
            {
                NHibernate.ICriteria qry = NHSession.CreateCriteria(typeof(Usuario) );
                usu = qry.List<Usuario>();
            }
            return usu;

        }

/// <summary>
/// LLENA LOS DATOS DEL USUARIO AL DBGRID
/// </summary>

       private void getTodosUsuarios()
        {

            this.dgvLista.DataSource = GetTUsuarios();
            if (dgvLista.RowCount == 0)
                lblCant.Text = "0";
            else
                lblCant.Text = dgvLista.RowCount.ToString();
        }


        private void btnCerrar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void FrmListarUsuarios_Load(object sender, EventArgs e)
        {
            getTodosUsuarios();
        }

        private void btnListar_Click(object sender, EventArgs e)
        {

            FrmNuevoUsuario objusu = new FrmNuevoUsuario();
            if (objusu.ShowDialog()== DialogResult.OK)
                getTodosUsuarios();
        }
    }
}

        public static IList<Usuario> GetAllUsuarioByID(String id)
        {
            ISession getNewS = clsPrincipal.GetNewSession();
            //obtiene la sesesion
            //List<Usuario> usuario;
           // Usuario usuario;
            using (NHibernate.ISession NHSession = getNewS)
            {
```

```
            IList<Usuario> user =
NHSession.CreateCriteria(typeof(Usuario)).Add(Expression.Like("UsuNombres", id,
MatchMode.Anywhere)).List<Usuario>();
            return user;
        }
    }

private void txtNombre_TextChanged(object sender, EventArgs e)
    {
        Usuario objusuario = new Usuario();
        if (txtNombre.Text == "")
            getTodosUsuarios();
        else
        getTodosUsuariosporNombre(txtNombre.Text.ToString());
    }
```

Paso 27.- Para realizar las búsquedas o filtros, se indicará una de las muchas alternativas para realizarlo, en la figura Nº III.59

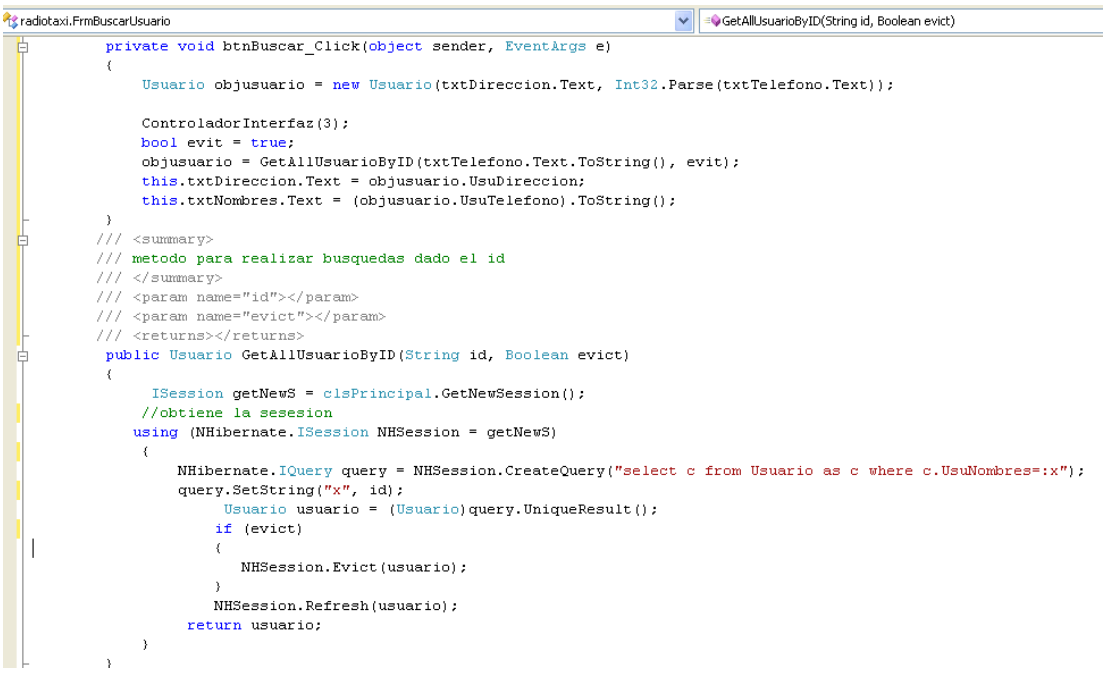


Figura Nº III. 59. Código para el método select By ID

A continuación se presenta el código que se encuentra en la figura Nº III.59

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
```

```
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;

namespace radiotaxi
{
   public partial class FrmBuscarUsuario : Form
   {
      public FrmBuscarUsuario()
      {
         InitializeComponent();
      }


      //variable del tipo Isession que se le asigna el valor obtenido de la session


      private void FrmBuscarUsuario_Load(object sender, EventArgs e)
      {
         ControladorInterfaz(1);
      }

      private void txtTelefono_Validating(object sender, CancelEventArgs e)
      {
         if ((this.txtTelefono.Text == "Ninguno") || (this.txtTelefono.Text == ""))//si no se ha ingresado datos
         { //MessageBox.Show("Ingrese el Número de Teléfono");
            e.Cancel = true;//cancelece Para q pase al siguiente control

            this.erpError.SetError(this.txtTelefono, "Ingreselos Nombres");//muestra un mensaje de error

         }
         else
            this.erpError.SetError(this.txtTelefono, "");
      }

      private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (Char.IsDigit(e.KeyChar) == true)//detecta si se ingresa letras
         {
            MessageBox.Show("No puede ingresar números", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
         }
      }


      private void txtNombres_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (Char.IsLetter(e.KeyChar) == true)//detecta si se ingresa numero
         {
            MessageBox.Show("No puede ingresar letras", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
         }
      }


      private void btnIniciar_Click(object sender, EventArgs e)
      {
         ControladorInterfaz(2);
```

```
        }


        private void btnOpcion_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(4);
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(5);
        }

        private void btnBuscar_Click(object sender, EventArgs e)
        {
            Usuario objusuario = new Usuario(txtDireccion.Text, Int32.Parse(txtTelefono.Text));

            ControladorInterfaz(3);
            bool evit = true;
            objusuario = GetAllUsuarioByID(txtTelefono.Text.ToString(), evit);
            this.txtDireccion.Text = objusuario.UsuDireccion;
            this.txtNombres.Text = (objusuario.UsuTelefono).ToString();
        }
        /// <summary>
        /// metodo para realizar busquedas dado el id
        /// </summary>
        /// <param name="id"></param>
        /// <param name="evict"></param>
        /// <returns></returns>
        public Usuario GetAllUsuarioByID(String id, Boolean evict)
        {
            ISession getNewS = clsPrincipal.GetNewSession();
            //obtiene la sesesion
            using (NHibernate.ISession NHSession = getNewS)
            {
                NHibernate.IQuery query = NHSession.CreateQuery("select c from Usuario as c where
c.UsuNombres=:x");
                query.SetString("x", id);
                    Usuario usuario = (Usuario)query.UniqueResult();
                if (evict)
                {
                    NHSession.Evict(usuario);
                }
                NHSession.Refresh(usuario);
                return usuario;
            }
        }

        private void btnCerrar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Paso 28.- Para realizar la actualización de datos, se procede a utilizar el siguiente código que se plantea como alternativa, el mismo que se muestra en la figura Nº III.60

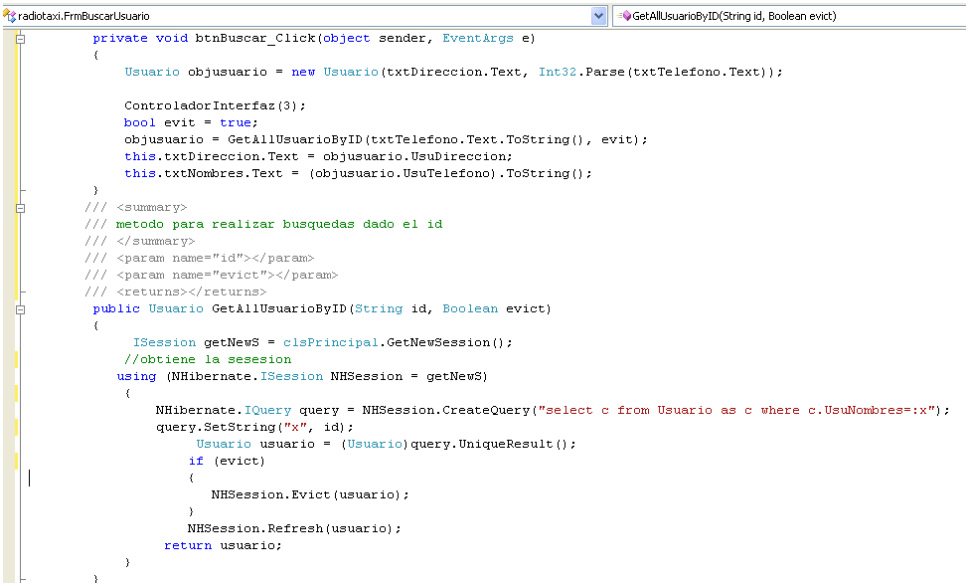


Figura Nº III. 60. Código para el método Update

Para detallar de mejor manera el código representado en la figura Nº III.60, se presenta a continuación, todo el código del formulario Actualizar Usuario

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;

namespace radiotaxi
{
   public partial class FrmBuscarUsuario : Form
   {
      public FrmBuscarUsuario()
      {
         InitializeComponent();
      }


      //variable del tipo Isession que se le asigna el valor obtenido de la session


      private void FrmBuscarUsuario_Load(object sender, EventArgs e)
      {
         ControladorInterfaz(1);
```

```
        }

        private void txtTelefono_Validating(object sender, CancelEventArgs e)
        {
            if ((this.txtTelefono.Text == "Ninguno") || (this.txtTelefono.Text == ""))//si no se ha ingresado datos
            { //MessageBox.Show("Ingrese el Número de Teléfono");
                e.Cancel = true;//cancelece Para q pase al siguiente control

                this.erpError.SetError(this.txtTelefono, "Ingreselos Nombres");//muestra un mensaje de error

            }
            else
                this.erpError.SetError(this.txtTelefono, "");
        }

        private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (Char.IsDigit(e.KeyChar) == true)//detecta si se ingresa letras
            {
                MessageBox.Show("No puede ingresar números", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
            }
        }


        private void txtNombres_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (Char.IsLetter(e.KeyChar) == true)//detecta si se ingresa numero
            {
                MessageBox.Show("No puede ingresar letras", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
            }
        }


        private void btnIniciar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(2);
        }


        private void btnOpcion_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(4);
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(5);
        }

        private void btnBuscar_Click(object sender, EventArgs e)
        {
            Usuario objusuario = new Usuario(txtDireccion.Text, Int32.Parse(txtTelefono.Text));

            ControladorInterfaz(3);
            bool evit = true;
            objusuario = GetAllUsuarioByID(txtTelefono.Text.ToString(), evit);
            this.txtDireccion.Text = objusuario.UsuDireccion;
            this.txtNombres.Text = (objusuario.UsuTelefono).ToString();
        }
       /// <summary>
```

```
        /// metodo para realizar busquedas dado el id
        /// </summary>
        /// <param name="id"></param>
        /// <param name="evict"></param>
        /// <returns></returns>
         public Usuario GetAllUsuarioByID(String id, Boolean evict)
        {
             ISession getNewS = clsPrincipal.GetNewSession();
            //obtiene la sesesion
           using (NHibernate.ISession NHSession = getNewS)
            {
               NHibernate.IQuery query = NHSession.CreateQuery("select c from Usuario as c where
c.UsuNombres=:x");
               query.SetString("x", id);
                    Usuario usuario = (Usuario)query.UniqueResult();
                  if (evict)
                  {
                    NHSession.Evict(usuario);
                  }
                  NHSession.Refresh(usuario);
                 return usuario;
            }
        }

public static void Actualizar(Usuario usuario)
        { //variable del tipo Isession que se le asigna el valor obtenido de la session
            ISession getNewS = clsPrincipal.GetNewSession();
            //obtiene la sesesion
            using (NHibernate.ISession NHSessionUsuario = getNewS)
            {
               //empieza la transaccion
               using (NHibernate.ITransaction tx = NHSessionUsuario.BeginTransaction())
               {
                  try
                  {
                     //guarda la sesion con los campos de la clase usuario
                     NHSessionUsuario.Update(usuario, usuario.UsuNombres);
                     tx.Commit();
                  }
                  catch
                  { tx.Rollback(); }
                  NHSessionUsuario.Close();
                  clsPrincipal.CloseSessionFactory();
               }
            }
        }

        private void btnCerrar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Paso 29.- Para la eliminación de información de un elemento, se emplea el método Delete, el mismo que se encuentra reflejado en la figura Nº III.61

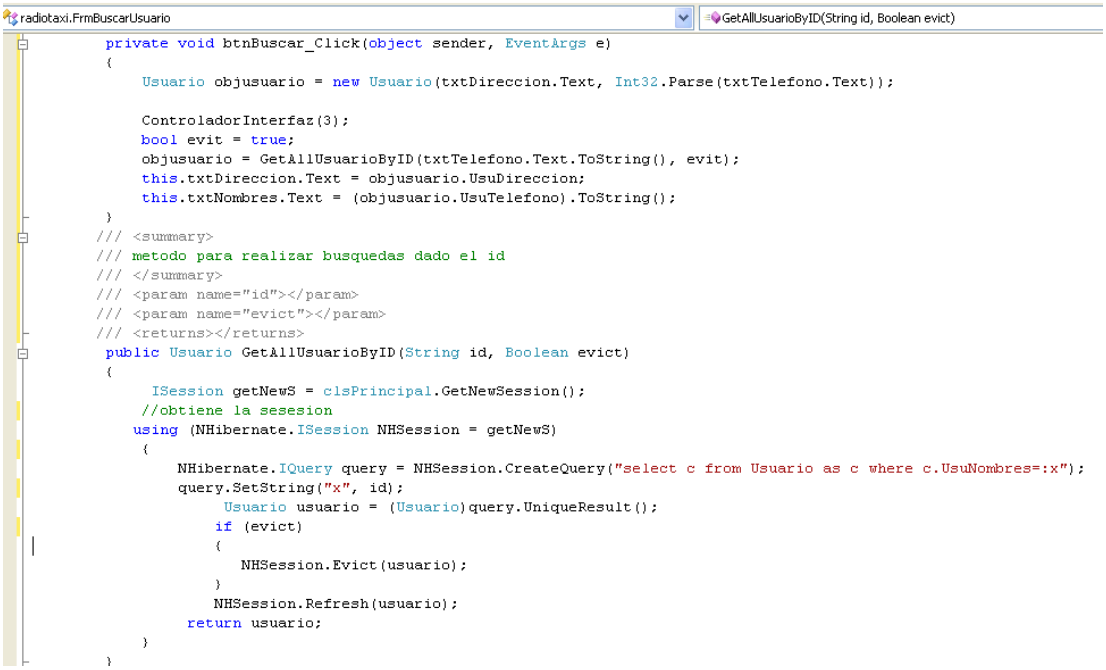


Figura Nº III. 61. Código para el método delete

A continuación se presenta el código que se encuentra en la figura Nº III.59

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;

namespace radiotaxi
{
   public partial class FrmBuscarUsuario : Form
   {
      public FrmBuscarUsuario()
      {
         InitializeComponent();
      }


      //variable del tipo Isession que se le asigna el valor obtenido de la session


      private void FrmBuscarUsuario_Load(object sender, EventArgs e)
      {
```

```
            ControladorInterfaz(1);
        }

        private void txtTelefono_Validating(object sender, CancelEventArgs e)
        {
            if ((this.txtTelefono.Text == "Ninguno") || (this.txtTelefono.Text == ""))//si no se ha ingresado datos
            { //MessageBox.Show("Ingrese el Número de Teléfono");
                e.Cancel = true;//cancelece Para q pase al siguiente control

                this.erpError.SetError(this.txtTelefono, "Ingreselos Nombres");//muestra un mensaje de error

            }
            else
                this.erpError.SetError(this.txtTelefono, "");
        }

        private void txtTelefono_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (Char.IsDigit(e.KeyChar) == true)//detecta si se ingresa letras
            {
                MessageBox.Show("No puede ingresar números", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
            }
        }


        private void txtNombres_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (Char.IsLetter(e.KeyChar) == true)//detecta si se ingresa numero
            {
                MessageBox.Show("No puede ingresar letras", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
            }
        }


        private void btnIniciar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(2);
        }


        private void btnOpcion_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(4);
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(5);
        }

        private void btnBuscar_Click(object sender, EventArgs e)
        {
            Usuario objusuario = new Usuario(txtDireccion.Text, Int32.Parse(txtTelefono.Text));

            ControladorInterfaz(3);
            bool evit = true;
            objusuario = GetAllUsuarioByID(txtTelefono.Text.ToString(), evit);
            this.txtDireccion.Text = objusuario.UsuDireccion;
            this.txtNombres.Text = (objusuario.UsuTelefono).ToString();
        }
```

```
        /// <summary>
        /// metodo para realizar busquedas dado el id
        /// </summary>
        /// <param name="id"></param>
        /// <param name="evict"></param>
        /// <returns></returns>
         public Usuario GetAllUsuarioByID(String id, Boolean evict)
         {
              ISession getNewS = clsPrincipal.GetNewSession();
             //obtiene la sesesion
            using (NHibernate.ISession NHSession = getNewS)
             {
                NHibernate.IQuery query = NHSession.CreateQuery("select c from Usuario as c where
c.UsuNombres=:x");
                query.SetString("x", id);
                     Usuario usuario = (Usuario)query.UniqueResult();
                   if (evict)
                   {
                     NHSession.Evict(usuario);
                   }
                   NHSession.Refresh(usuario);
                 return usuario;
             }
         }

         private void btnCerrar_Click(object sender, EventArgs e)
         {
            this.Close();
         }
     }
}



public Usuario Eliminar(String nombre)
        { //variable del tipo Isession que se le asigna el valor obtenido de la session
            ISession getNewS = clsPrincipal.GetNewSession();
            Usuario usuario;
            //obtiene la sesesion
            using (NHibernate.ISession NHSessionUsuario = getNewS)
            {
                //empieza la transaccion
                using (NHibernate.ITransaction tx = NHSessionUsuario.BeginTransaction())
                {

                     NHibernate.IQuery query = NHSessionUsuario.CreateQuery("select c from Usuario as c
where c.UsuNombres=:x");
                query.SetString("x", nombre);
                usuario = (Usuario)query.UniqueResult();
                       //guarda la sesion con los campos de la clase usuario
                       NHSessionUsuario.Delete(usuario);
                       tx.Commit();

                  /*  }
                    catch
                    { tx.Rollback(); }
                    NHSessionUsuario.Close();
                    clsPrincipal.CloseSessionFactory();
                    */return usuario;
                }

            }

        }
```

Paso 30.- Para generar reportes a través de Crystal Report, se debe crear el respectivo archivo data set, en el que se deberá crear los DataTables de acuerdo a los existentes en la base de datos, después se crea el cristal report viewer, y lo configuramos con el documento de cristal report, como se presenta en la figura Nº III.62, en la que se muestra el código indicando la configuración para el dataset, la clase, y la presentación de los datos.

ListarUsuario.cs | ListarUsuario.xsd | hibernate.cfg.xml | Program.cs | CrystalReport1.rpt | FrmPrincipal.cs | FrmPrincipal.cs [Diseño] | FrmRptUsuarios.cs*

radiotaxi.FrmRptUsuarios | cargaReport()

```
        private void cargaReport()
        {
            try
            {
                Usuario objusuario = new Usuario();
                ListarUsuario dsusuarios = new ListarUsuario();
                ISession getNewS = clsPrincipal.GetNewSession();
                   using (NHibernate.ISession NHSession = getNewS)
                {
                    IList<Usuario> user = NHSession.CreateCriteria(typeof(Usuario)).List<Usuario>();
                    foreach (Usuario u in user)
                    {
                        DataRow fila = dsusuarios.Tables["Usuario"].NewRow();
                        fila["usu_nombres"] = u.UsuNombres;
                        fila["usu_direccion"] = u.UsuDireccion;
                        fila["usu_telefono"] = u.UsuTelefono;
                    }
                    dsusuarios.Tables["Usuario"].AcceptChanges();
                   // CrystalReport1 rpt = new CrystalReport1();
                    REPORTES.rptUsuarios rpt= new radiotaxi.REPORTES.rptUsuarios();
                    rvUsuario.ReportSource = rpt;
                    rvUsuario.Refresh();
                }
            }
            catch
            { MessageBox.Show("Error"); }

        }


        private void FrmRptUsuarios_Load(object sender, EventArgs e)
        {
            cargaReport();
```

Figura Nº III. 62. Código para reportes de Crystal Report

Para respaldar el código presentado en la figura Nº III.62, se lista el código que contiene el formulario con el cristal report viewer.

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;


namespace radiotaxi
{
   public partial class FrmRptUsuarios : Form
   {
      public FrmRptUsuarios()
      {
         InitializeComponent();
      }

      private void cargaReport()
      {
         try
         {
            Usuario objusuario = new Usuario();
            ListarUsuario dsusuarios = new ListarUsuario();
            ISession getNewS = clsPrincipal.GetNewSession();
              using (NHibernate.ISession NHSession = getNewS)
            {
               IList<Usuario> user = NHSession.CreateCriteria(typeof(Usuario)).List<Usuario>();
               foreach (Usuario u in user)
               {
                  DataRow fila = dsusuarios.Tables["Usuario"].NewRow();
                  fila["usu_nombres"] = u.UsuNombres;
                  fila["usu_direccion"] = u.UsuDireccion;
                  fila["usu_telefono"] = u.UsuTelefono;
               }
               dsusuarios.Tables["Usuario"].AcceptChanges();
              // CrystalReport1 rpt = new CrystalReport1();
               REPORTES.rptUsuarios rpt= new radiotaxi.REPORTES.rptUsuarios();
               rvUsuario.ReportSource = rpt;
               rvUsuario.Refresh();
            }
         }
         catch
         { MessageBox.Show("Error"); }

      }


      private void FrmRptUsuarios_Load(object sender, EventArgs e)
      {
         cargaReport();
      }
   }
}
```

CAPITULO IV

DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA PARA LA DETECCIÓN DE LLAMADAS

Para realizar el desarrollo e implementación del sistema para la detección de llamadas, se estableció seguir la Metodología Orientada a Objetos de Microsoft Solution Framework MSF, la misma que se centra específicamente al desarrollo de las actividades específicas del proyecto a realzarse, puesto que fomenta la reutilización de componentes y clases existentes, por lo se disminuye el tiempo, la programación se facilita, los costos de desarrollo bajan, los recursos disminuyen y puede ser adaptable a cualquier tipo de proyecto ya sea de escritorio o web, en el que en cada etapa o fase se identifica las actividades concretas a desarrollarse para cumplir la programación e implementación de nuestro proyecto para la detección de llamadas, facilitando la ejecución de las tareas en el desarrollo y cumplimiento de las mismas.

4.1. DEFINICIÓN

En ésta primera fase se realizará el estudio del problema a resolver indicando de antemano, las necesidades a resolver, ejecutando el planteamiento de posibles soluciones, acuerdos, tecnologías, etc., el tiempo y costos estimados para el desarrollo y ejecución de la solución la misma que cubrirá todas las expectativas de los usuarios del sistema planteado como necesidad o requerimientos

4.1.1. Problema

Las actividades que se desarrollan actualmente en la Cooperativa de taxis TAXIPHONE, vienen desarrollándose con ciertas deficiencias, por tal motivo se requiere automatizar las tareas administrativas en relación al control de llamadas telefónicas de los señores usuarios, información de los taxistas existentes, registros y reportes de las carreras atendidas y de taxistas afiliados a la cooperativas.

El proceso de trabajo que actualmente se desarrolla en la Cooperativa, no se obtiene los resultados esperados ya que la información que se maneja es solo a través de papeles lo que se dificulta tener un control exacto de la información que se maneja.

Se requiere un sistema informático que proporcione facilidad al manejo de la información que por lo general lo realiza la secretaría u operadora; la información que se almacena es de los usuarios, de los señores taxistas, de las carreras realizadas en el día, de las llamadas atendidas, reportes de las carreras que ha realizado un taxista,

cantidad de taxis que laboran en el día, llamadas atendidas, etc, en base a esta información se podrá conocer realmente el servicio de la cooperativa hacia la ciudadanía y más aun conocer detenidamente el desplazamiento de los taxis en la ciudad con la atención de las distintas carreras

4.1.2. Visión del Proyecto

Otorgar estabilidad y confiabilidad de información, que ayudarán para la toma de decisiones en el área administrativa con el control de usuarios y de socios taxistas de la cooperativa, facilitando el manejo y organización de la información, obteniendo acceso a la misma de una manera rápida y confiable.

4.1.3. Perfiles de Usuario

Los potenciales usuarios que podría tener el sistema informático son:

- ✓ La operadora o secretaria, la misma que se encarga de atender las llamadas telefónicas, registrar a usuarios y taxis, realizar búsquedas, actualizaciones, eliminaciones, generará reportes necesarios
- ✓ El presidente de la cooperativa, podrá acceder a la visualización de la información y a reportes necesarios

4.1.4. Ámbito del Proyecto

El ámbito del proyecto está definido por los requerimientos o necesidades del cliente, los mismos que a continuación se detalla:

- ✓ El usuario llama a la central a solicitar un taxi; el usuario proporciona su nombre y apellidos con la dirección exacta en donde reside
- ✓ La secretaria toma los datos del usuario necesarios para enviar el taxi a la dirección especificada.
- ✓ La secretaria informa a todos los taxis que están laborando a través del radio satelital, que hay una carrera en espera para que sea atendida por uno de ellos que se encuentra disponible
- ✓ Una vez que se detecta la disponibilidad del taxi, se le proporciona toda la información del usuario; el taxista llega al destino indicado, y procede a reportar a la secretaria que ha realizado la carrera con el destino de la misma
- ✓ La secretaria registra como carrera atendida con los datos del taxi, del usuario y de la carrera con la hora y destino.
- ✓ La secretaria empieza a ubicarles a todos los taxis que se encuentran laborando, realizando llamadas de rastreo, dando a conocer el número de taxi y el destino en el cual se encuentra

Además se debe reportes de los taxis que laboran en el día, dando a conocer la hora de inicio de la labor y la hora que termina su labor en el día

Se debe tener reportes de la siguiente con la siguiente información:

- ✓ Descripción de los taxis que hayan laborado en el día

- ✓ Registro de todas las llamadas recibidas a la central de la cooperativa solicitando un taxi
- ✓ Registro de las llamadas atendidas
- ✓ Registro de todos los taxis existentes en la cooperativa
- ✓ Informe de todos los usuarios registrados en la cooperativa.

A continuación se describe la lista de los principales requerimientos para el sistema informático a desarrollarse:

Requerimiento 1: Ingreso, actualización, eliminación y listado de los usuarios

Requerimiento 2: Ingreso, actualización, eliminación y listado de los taxis

Requerimiento 3: Ingreso, listado de las carreras

Requerimiento 4: Registro, listados, búsquedas de llamadas

Requerimiento 5: Cambio de estado de los taxis

Requerimiento 6: Cambio de disponibilidad de taxis

Requerimiento 7: Reportes de carreras, taxis, llamadas, usuarios

4.1.5. Concepto de Solución

La solución planteada para la automatización de las actividades de la cooperativa Taxiphone, están enmarcadas a facilitar las tareas aquí realizadas, por lo que se aprovechará los equipos hardware con que cuenta la cooperativa.

Al sistema para la detección y control de llamadas se lo llamará sistema Taxiphone, que su fin u objetivo será proporcionar un identificador de llamadas, motivo por el cual

se necesitará que el equipo telefónico se encuentre conectado a la computadora, además de ello para que la secretaria pueda informar a los señores taxistas de las carreras solicitadas se utilizará el radio satelital con el que cuenta la cooperativa.

El sistema se lo desarrollará en la plataforma Windows XP con las herramientas de desarrollo de Microsoft Visual Studio .net 2005 empleando el framework NHibernate, utilizando herramientas de apoyo como Microsoft Visio, Project 2007, para el manejo y administración de la base de datos estará en el motor de base de datos SQL Server 2005, para los reportes se utilizará Crystal Report.

En la figura Nº IV.63, se esquematiza el modelo solución planteado con sus características técnicas, en el que se indica el proceso de las actividades de la cooperativa taxiphone, iniciando desde la llamada telefónica del usuario solicitando el servicio de taxi, la misma que será detectada por el sistema indicando el nombre, dirección y teléfono de dicho usuario que será atendida por la secretaria, reportando inmediatamente a todos los taxistas que se encuentran laborando, para que uno de ellos asista a la carrera solicitada

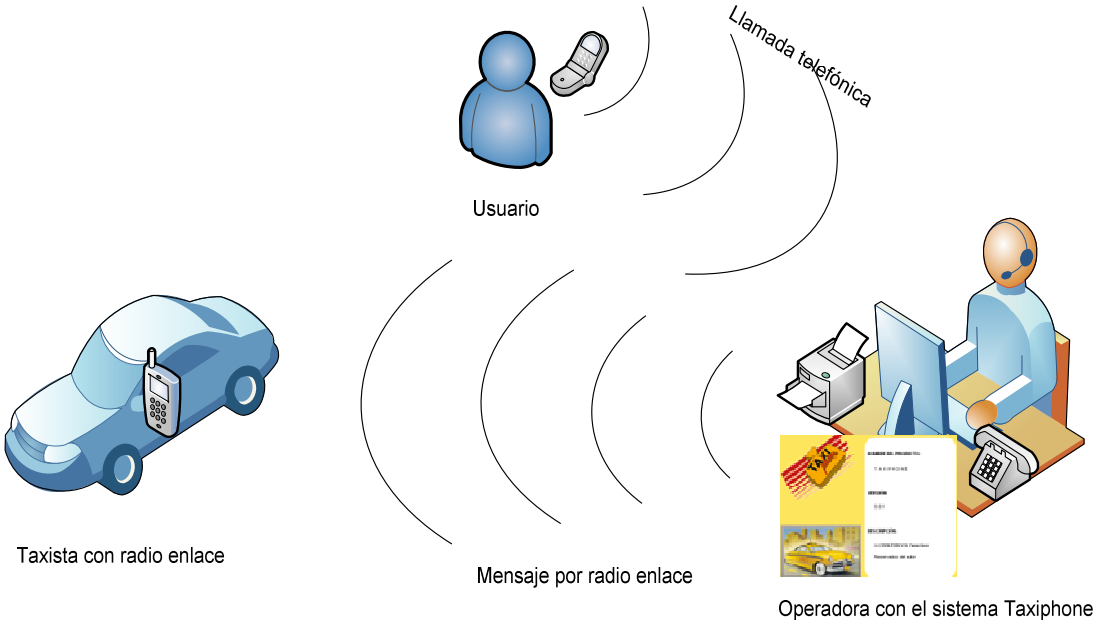


Figura Nº IV. 63. Solución planteada para la Cooperativa Taxiphone

A continuación se presenta la Figura Nº IV.64, indicando las características técnicas de desarrollo con las que se pretende realizar el sistema, en el que se establece las herramientas Microsoft como herramientas case que servirán de apoyo, se utiliza además Visual Studio .net 2005 con el lenguaje de programación C# como herramienta de desarrollo empleando el framework NHibernate interactuando con el motor de base de datos SQL Server 2005 y para los reportes requeridos se empleará Crystal Report

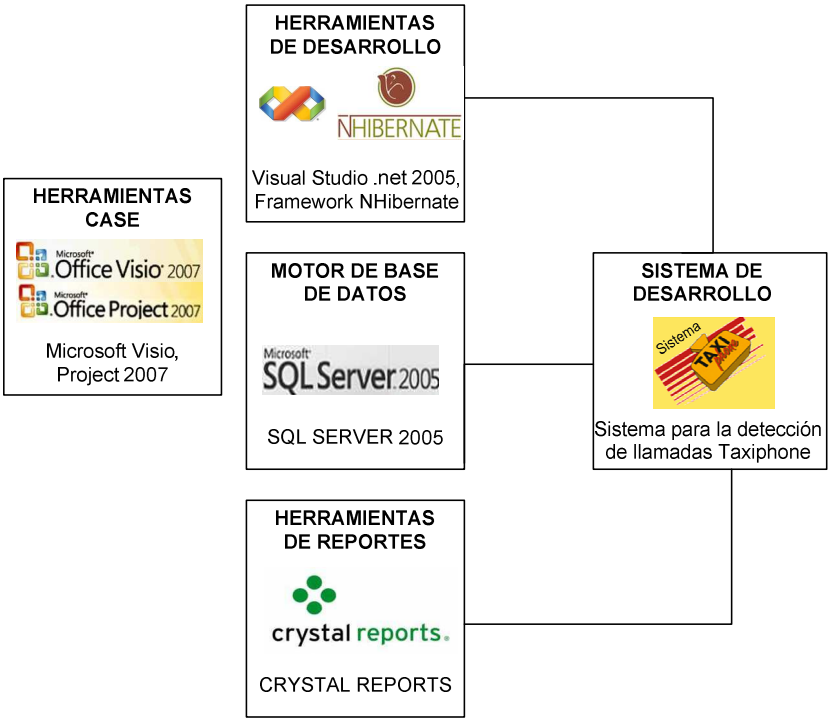


Figura Nº IV. 64. Características técnicas de desarrollo del sistema taxiphone

4.1.6. Objetivos del Proyecto

Se listarán los objetivos que se cumplirán en el ámbito del negocio como en el diseño, los mismo que proporcionarán estabilidad entre las partes, estableciendo acuerdos y otorgamientos de beneficios, con la adquisición del sistema.

4.1.6.1. Objetivos del Negocio

Los objetivos del proyectos están establecidos como las metas que se quieren conseguir con el desarrollo del sistema TAXIPHONE otorgando beneficios en cuanto al desarrollo de las actividades de la cooperativa, que se presentan gracias al diseño del sistema

basado en herramientas hardware y software adecuados que proporcione la explotación al máximo del sistema.

4.1.6.2. Objetivos del Diseño

- ✓ Vincular las herramientas de desarrollo C#.NET con el framework NHibernate generando los archivos de configuración que permitan interactuar con la base de datos, ofreciendo estabilidad en el manejo de la información

- ✓ Otorgar confiabilidad en la información obtenida, la misma que será almacenada en la base de datos respectiva

- ✓ Proporcionar un ambiente amigable y seguro en el manejo e interacción con el sistema TAXIPHONE.

4.1.7. Factores Críticos

Debido al ámbito al que están enfocados todos los proyectos software, existen ciertos factores críticos o riesgos que pueden ser considerados como genéricos, en el sentido en que pueden presentarse en una solución software independientemente del entorno en el cual vaya a ser desplegada.

A continuación se presenta la tabla Nº IV. 3 que hace referencia a este tipo de riesgos, que serán considerados y evaluados también para el desarrollo de nuestro sistema taxiphone.

Tabla Nº IV. 3. Tabla de factores de Riesgo

Código	Riesgo
R1	Software elegido para el desarrollo de la aplicación no presenta todas las facilidades para solventar las necesidades exigidas por el usuario.
R2	Falta de experiencia del personal para desarrollar con la herramienta elegida para dicho fin
R3	Un evento inesperado produce una ruptura organizacional durante el período de ejecución del proyecto.
R4	Ausencia o falta de capital para solventar la ejecución del proyecto
R5	Requerimiento de un usuario, mal entendido desde un inicio por parte del desarrollador
R6	Número más grande de requerimientos que los inicialmente planteados
R7	Las especificaciones de las interfaces principales no estarán a tiempo para la ejecución de las actividades de diseño.
R8	La estimación de los costos de desarrollo de la aplicación ha sido inferior a la realmente necesitada.
R9	El tamaño del software es subestimado
R10	Desinterés para el desarrollo en momentos críticos
R11	Reajuste de la agenda de trabajo.
R12	Fuente de información ambigua
R13	Fallas técnicas en el hardware usado para el desarrollo de la aplicación

Categorización de riesgos

Estimación de la probabilidad y consecuencias de los riesgos

Luego de haber definido los posibles riesgos que pueden presentarse dentro de la ejecución del sistema, se establece la probabilidad de ocurrencia de cada uno de ellos así como los efectos que causarán en caso de presentarse.

En la tabla Nº IV.4, se resume los posibles problemas a presentarse y la valoración de impacto de cada uno de ellos

Tabla Nº IV. 4. Categorización de Riesgos

Código Riesgo	Efectos			Probabilidad que ocurra		
	Tolerable	Serio	Catastrófico	Bajo	Medio	Alto
R1	X			X		
R2		X				X
R3			X	X		
R4		X				X
R5	X				X	
R6	X					X
R7	X			X		
R8			X			X
R9	X				X	
R10		X				X

R11			**X**		**X**	
R12		**X**			**X**	
R13			**X**		**X**	

Planes de contingencia.

Luego de haber realizado la evolución de los posibles riesgos que puedan presentarse en el período de desarrollo del sistema Taxiphone, se ha previsto posibles planes de contingencia enfocados específicamente a nuestro problema y riesgo, presentados en la tabla Nº IV.5.

A pesar de que existen estándares para la ejecución de planes de contingencia hemos querido particularizarlos para el entorno y circunstancias que presenta nuestra solución y lo analizaremos los 10 riesgos potenciales que se podrían presentar

Tabla Nº IV. 5. Plan de Contingencia de los Riesgos

Riesgo	Estrategia
Software elegido para el desarrollo de la aplicación no presenta todas las facilidades para solventar las necesidades exigidas por el usuario.	Antes de empezar el desarrollo realizar un estudio comparativo entre todos los posibles software de desarrollo, especificando las características de cada uno de ellos, y se elegirá el que mejor se adapte a las características del sistema que se desea desarrollar.

Falta de experiencia del personal para desarrollar con la herramienta elegida para dicho fin	Para seleccionar como herramienta de desarrollo se deberá tomar en consideración el conocimiento o el manejo de herramientas de desarrollo que no compliquen más bien ayude al desarrollador a cumplir su objetivo
Un evento inesperado produce una ruptura organizacional durante el período de ejecución del proyecto.	Reasignar tareas, estableciendo la capacidad de cada miembro del equipo para realizar una tarea determinada. Otra opción sería contratar personas externas al sistema que tengan experiencia para que se hagan cargo del módulo que quedó sin personal
Requerimiento de un usuario, mal entendido desde un inicio por parte del desarrollador	Realizar un análisis completo acerca del impacto que tendría en la versión actual del sistema, la inclusión de los nuevos requerimientos, estimar la posibilidad de adecuar el sistema a ellos.
Número más grande de requerimientos que los inicialmente planteados	Realizar una reestructuración en la planificación y estimación anterior, estableciendo los cambios que va a sufrir el sistema, sin que se presente inconvenientes en el desarrollo del sistema
Las especificaciones de las interfaces principales no estarán a tiempo para la ejecución de las actividades de diseño.	En lo posible aplicar y cumplir el cronograma de actividades, de no ser así, realizar un ajuste en las actividades a desarrollarse, sin que afecte el tiempo de entrega del sistema
La estimación de los costos	Desarrollar una nueva estimación de costos basándonos

de desarrollo de la aplicación ha sido inferior a la realmente necesitada	en el desarrollo real del sistema, proporcionando al cliente, la información por la que ha cambiado la estimación del costo, sin que se rompa el trato de desarrollo
Desinterés para el desarrollo en momentos críticos	Proporcionar nuevas técnicas de trabajo que motiven al personal encargado del desarrollo el seguimiento del mismo, hasta llegar a su etapa de liberación
Reajuste de la agenda de trabajo.	Realizar un nuevo cronograma de trabajo estimando posibles problemas en tiempo de desarrollo evitando múltiples ajuste de agenda de trabajo, más bien coordinar con el personal de desarrollo, intentando cumplir con el cronograma establecido
Fallas técnicas en el hardware usado para el desarrollo de la aplicación	Proveer de equipos o dispositivos buenos que no dificulten el desarrollo del sistema, más bien faciliten el cumplimiento de la entrega a tiempo del sistema

4.1.8. Planificación Inicial

A continuación se presenta la Figura Nº IV.65 y en la Figura Nº IV.66, indicando el cronograma de actividades que se ejecutarán para el desarrollo del sistema Taxiphone, dichas cumplen las cinco fases de la metodología **MSF** (Microsoft Solution Framework) las cuales son desempeñadas por los roles que intervienen en la misma metodología, en la fase de Definición se emplearán 37 días empezando desde el Lunes 05 de Enero del 2009 hasta el Martes 24 de Febrero del 2009 ejecutada por el Gerente del producto; en

la fase de Planificación se emplearán 96 días desde el miércoles 25 de Febrero del 2009 hasta el miércoles 08 de Julio del 2009 tareas que son ejecutadas por el Gerente del Programa y el Desarrollador; la fase de Desarrollo tiene una duración de 67 días empezando desde el viernes 03 de Julio del 2009 hasta el Lunes 05 de Octubre, éstas tareas que será, ejecutadas por el Desarrollador; en la fase de Estabilización se emplean 69 días que empezarían desde el Lunes 05 de Octubre del 2009 hasta el jueves 07 de Enero del 2010, éstas tareas serán realizadas por el Educador a usuarios y de Pruebas; la última fase de Instalación se realizará en 5 días que empezarían desde el Lunes 04 de Enero del 2010 hasta el viernes 08 de Enero del 2010, en total éstas tareas tiene una duración de 270 días laborables

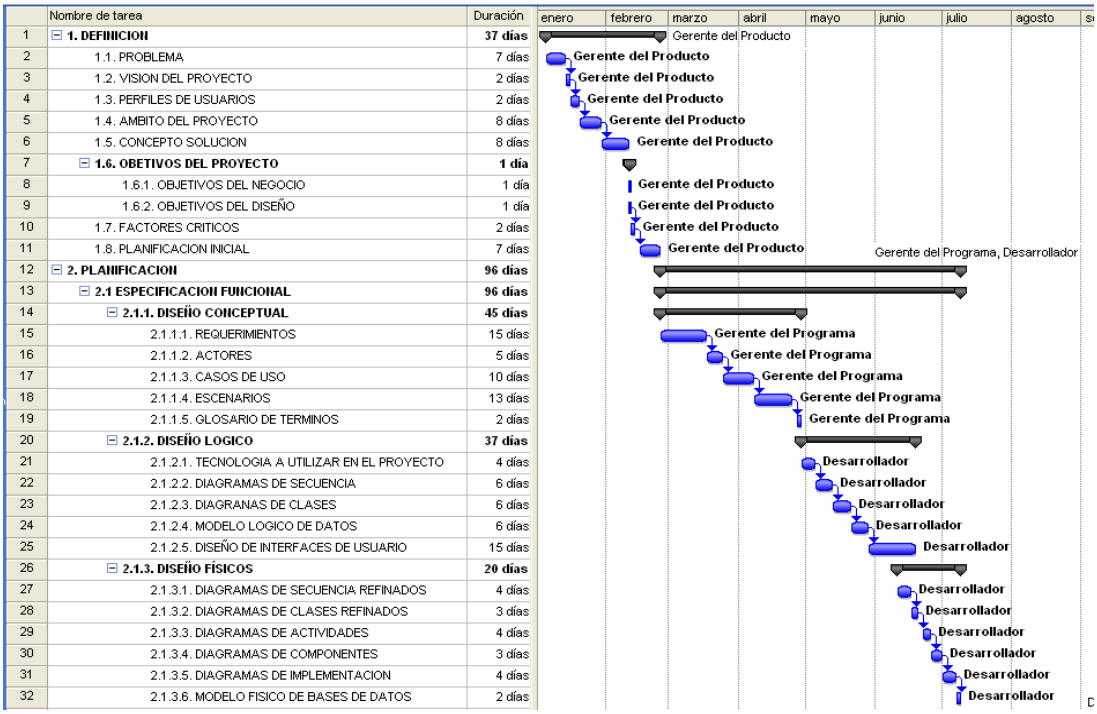
	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1	**1. DEFINICION**	**37 días**	**lun 05/01/09**	**mar 24/02/09**	**Gerente del Producto**
2	1.1. PROBLEMA	7 días	lun 05/01/09	mar 13/01/09	Gerente del Producto
3	1.2. VISION DEL PROYECTO	2 días	mié 14/01/09	jue 15/01/09	Gerente del Producto
4	1.3. PERFILES DE USUARIOS	2 días	vie 16/01/09	lun 19/01/09	Gerente del Producto
5	1.4. AMBITO DEL PROYECTO	8 días	mar 20/01/09	jue 29/01/09	Gerente del Producto
6	1.5. CONCEPTO SOLUCION	8 días	vie 30/01/09	mar 10/02/09	Gerente del Producto
7	**1.6. OBETIVOS DEL PROYECTO**	**1 día**	**mié 11/02/09**	**mié 11/02/09**	**Gerente del Producto**
8	1.6.1. OBJETIVOS DEL NEGOCIO	1 día	mié 11/02/09	mié 11/02/09	Gerente del Producto
9	1.6.2. OBJETIVOS DEL DISEÑO	1 día	mié 11/02/09	mié 11/02/09	Gerente del Producto
10	1.7. FACTORES CRITICOS	2 días	jue 12/02/09	vie 13/02/09	Gerente del Producto
11	1.8. PLANIFICACION INICIAL	7 días	lun 16/02/09	mar 24/02/09	Gerente del Producto
12	**2. PLANIFICACION**	**96 días**	**mié 25/02/09**	**mié 08/07/09**	**Gerente del Programa;Desarrollador**
13	**2.1 ESPECIFICACION FUNCIONAL**	**96 días**	**mié 25/02/09**	**mié 08/07/09**	**Gerente del Programa**
14	**2.1.1. DISEÑO CONCEPTUAL**	**45 días**	**mié 25/02/09**	**mar 28/04/09**	**Gerente del Programa**
15	2.1.1.1. REQUERIMIENTOS	15 días	mié 25/02/09	mar 17/03/09	Gerente del Programa
16	2.1.1.2. ACTORES	5 días	mié 18/03/09	mar 24/03/09	Gerente del Programa
17	2.1.1.3. CASOS DE USO	10 días	mié 25/03/09	mar 07/04/09	Gerente del Programa
18	2.1.1.4. ESCENARIOS	13 días	mié 08/04/09	vie 24/04/09	Gerente del Programa
19	2.1.1.5. GLOSARIO DE TERMINOS	2 días	lun 27/04/09	mar 28/04/09	Gerente del Programa
20	**2.1.2. DISEÑO LOGICO**	**37 días**	**mié 29/04/09**	**jue 18/06/09**	**Desarrollador**
21	2.1.2.1. TECNOLOGIA A UTILIZAR EN EL PROYECTO	4 días	mié 29/04/09	lun 04/05/09	Desarrollador
22	2.1.2.2. DIAGRAMAS DE SECUENCIA	6 días	mar 05/05/09	mar 12/05/09	Desarrollador
23	2.1.2.3. DIAGRANAS DE CLASES	6 días	mié 13/05/09	mié 20/05/09	Desarrollador
24	2.1.2.4. MODELO LOGICO DE DATOS	6 días	jue 21/05/09	jue 28/05/09	Desarrollador
25	2.1.2.5. DISEÑO DE INTERFACES DE USUARIO	15 días	vie 29/05/09	jue 18/06/09	Desarrollador
26	**2.1.3. DISEÑO FÍSICOS**	**20 días**	**jue 11/06/09**	**mié 08/07/09**	**Desarrollador**
27	2.1.3.1. DIAGRAMAS DE SECUENCIA REFINADOS	4 días	jue 11/06/09	mar 16/06/09	Desarrollador
28	2.1.3.2. DIAGRAMAS DE CLASES REFINADOS	3 días	mié 17/06/09	vie 19/06/09	Desarrollador
29	2.1.3.3. DIAGRAMAS DE ACTIVIDADES	4 días	lun 22/06/09	jue 25/06/09	Desarrollador
30	2.1.3.4. DIAGRAMAS DE COMPONENTES	3 días	vie 26/06/09	mar 30/06/09	Desarrollador
31	2.1.3.5. DIAGRAMAS DE IMPLEMENTACION	4 días	mié 01/07/09	lun 06/07/09	Desarrollador
32	2.1.3.6. MODELO FISICO DE BASES DE DATOS	2 días	mar 07/07/09	mié 08/07/09	Desarrollador

Figura Nº IV. 65. Diagrama de Actividades y asignación de recursos

	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
32	2.1.3.6. MODELO FISICO DE BASES DE DATOS	2 días	mar 07/07/09	mié 08/07/09	Desarrollador
33	**3. DESARROLLO**	**67 días**	**vie 03/07/09**	**lun 05/10/09**	**Desarrollador**
34	3.1. NOMENCLATURA Y ESTANDARES PARA EL DESARROLLO	3 días	vie 03/07/09	mar 07/07/09	Desarrollador
35	**3.2. CAPA DE PRESENTACION**	**20 días**	**mié 08/07/09**	**mar 04/08/09**	**Desarrollador**
36	3.2.1. DISEÑO DE INTERFACES DE USUARIO	15 días	mié 08/07/09	mar 28/07/09	Desarrollador
37	3.2.2. DISEÑO DE COMPONENTES DE PROCESAMIENTO DE US	5 días	mié 29/07/09	mar 04/08/09	Desarrollador
38	**3.3. CAPA DE DATOS**	**39 días**	**mié 29/07/09**	**lun 21/09/09**	**Desarrollador**
39	3.3.1. IMPLEMENTACION DE LA BASE DE DATOS	5 días	mié 29/07/09	mar 04/08/09	Desarrollador
40	3.3.2. IMPLEMENTACION DE ACCESO A DATOS	12 días	mié 05/08/09	jue 20/08/09	Desarrollador
41	3.3.3. DISEÑO DE ESQUEMAS	22 días	vie 21/08/09	lun 21/09/09	Desarrollador
42	**3.4. CAPA DE NEGOCIOS**	**4 días**	**lun 21/09/09**	**jue 24/09/09**	**Desarrollador**
43	3.4.1. IMPLEMENTACION DE COMPONENTES	4 días	lun 21/09/09	jue 24/09/09	Desarrollador
44	**3.5. ESPECIFICACION DE SEGURIDAD**	**7 días**	**vie 25/09/09**	**lun 05/10/09**	**Desarrollador**
45	3.5.1. DISEÑO DE ESTRATEGIAS DE AUTORIZACION, AUTEN	7 días	vie 25/09/09	lun 05/10/09	Desarrollador
46	**4. ESTABILIZACION**	**69 días**	**lun 05/10/09**	**jue 07/01/10**	**Educación a usuarios;Pruebas**
47	**4.1. REVISION GENERAL DEL SISTEMA**	**54 días**	**lun 05/10/09**	**jue 17/12/09**	**Educación a usuarios;Pruebas**
48	4.1.1. CODIGO FUENTE Y EJECUTABLE	12 días	lun 05/10/09	mar 20/10/09	Educación a usuarios
49	4.1.2. SCRIPTS DE BASE DE DATOS	1 día	mié 21/10/09	mié 21/10/09	Educación a usuarios
50	4.1.3. DOCUMENTACION DE LA INSTALACION	13 días	jue 22/10/09	lun 09/11/09	Educación a usuarios
51	4.1.4. AYUDA	13 días	mar 10/11/09	jue 26/11/09	Educación a usuarios
52	4.1.5. MATERIAL DE CAPACITACION	12 días	vie 27/11/09	lun 14/12/09	Educación a usuarios
53	4.1.6. HISTORIAL DE VERSIONES	3 días	mar 15/12/09	jue 17/12/09	Educación a usuarios
54	4.2. PRUEBAS	15 días	vie 18/12/09	jue 07/01/10	Pruebas
55	**5. INSTALACION**	**5 días**	**lun 04/01/10**	**vie 08/01/10**	**Gerente de Implantación;Educación a usu**
56	5.1. PLAN DE OPERACIÓN Y SOPORTE	2 días	lun 04/01/10	mar 05/01/10	Gerente de Implantación
57	5.2. PROCEDIMINETOS Y PROCESOS	1 día	mié 06/01/10	mié 06/01/10	Gerente de Implantación
58	5.3. HISTORIAL DEL SOPORTE TECNICO	2 días	jue 07/01/10	vie 08/01/10	Educación a usuarios

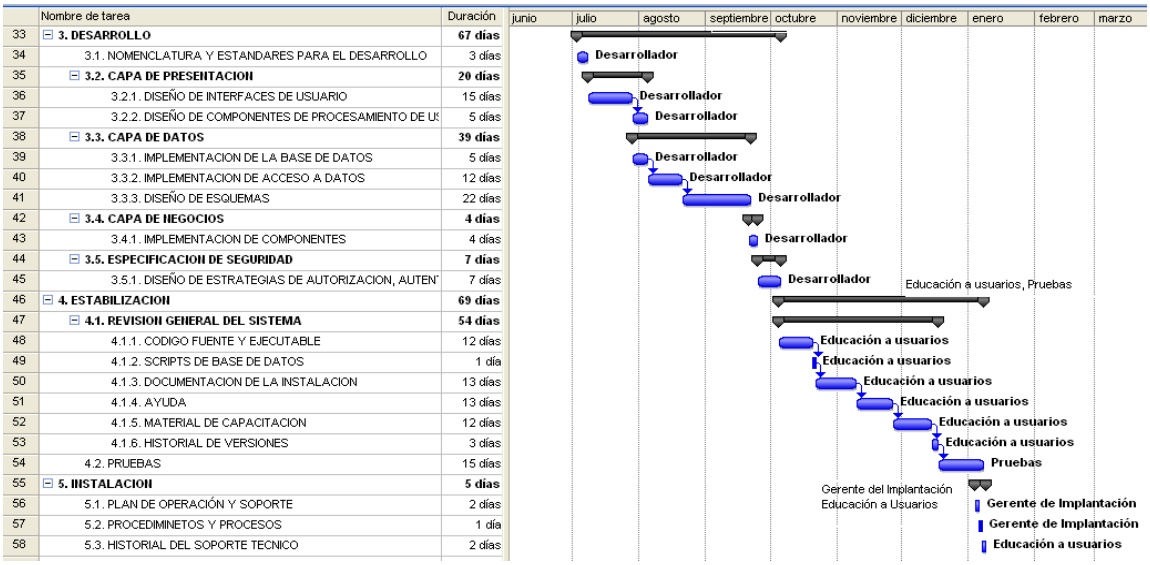
Figura Nº IV. 66. Diagramas de las últimas Actividades y asignación de recursos

En la figura Nº IV.67 y en la figura Nº IV.68, se ilustra el tiempo empleado en las actividades de desarrollo del sistema Taxiphone, con la utilización de los recursos en cada uno de ellos



	Nombre de tarea	Duración
1	**1. DEFINICION**	**37 días**
2	1.1. PROBLEMA	7 días
3	1.2. VISION DEL PROYECTO	2 días
4	1.3. PERFILES DE USUARIOS	2 días
5	1.4. AMBITO DEL PROYECTO	8 días
6	1.5. CONCEPTO SOLUCION	8 días
7	**1.6. OBETIVOS DEL PROYECTO**	**1 día**
8	1.6.1. OBJETIVOS DEL NEGOCIO	1 día
9	1.6.2. OBJETIVOS DEL DISEÑO	1 día
10	1.7. FACTORES CRITICOS	2 días
11	1.8. PLANIFICACION INICIAL	7 días
12	**2. PLANIFICACION**	**96 días**
13	**2.1 ESPECIFICACION FUNCIONAL**	**96 días**
14	**2.1.1. DISEÑO CONCEPTUAL**	**45 días**
15	2.1.1.1. REQUERIMIENTOS	15 días
16	2.1.1.2. ACTORES	5 días
17	2.1.1.3. CASOS DE USO	10 días
18	2.1.1.4. ESCENARIOS	13 días
19	2.1.1.5. GLOSARIO DE TERMINOS	2 días
20	**2.1.2. DISEÑO LOGICO**	**37 días**
21	2.1.2.1. TECNOLOGIA A UTILIZAR EN EL PROYECTO	4 días
22	2.1.2.2. DIAGRAMAS DE SECUENCIA	6 días
23	2.1.2.3. DIAGRANAS DE CLASES	6 días
24	2.1.2.4. MODELO LOGICO DE DATOS	6 días
25	2.1.2.5. DISEÑO DE INTERFACES DE USUARIO	15 días
26	**2.1.3. DISEÑO FÍSICOS**	**20 días**
27	2.1.3.1. DIAGRAMAS DE SECUENCIA REFINADOS	4 días
28	2.1.3.2. DIAGRAMAS DE CLASES REFINADOS	3 días
29	2.1.3.3. DIAGRAMAS DE ACTIVIDADES	4 días
30	2.1.3.4. DIAGRAMAS DE COMPONENTES	3 días
31	2.1.3.5. DIAGRAMAS DE IMPLEMENTACION	4 días
32	2.1.3.6. MODELO FISICO DE BASES DE DATOS	2 días

Figura Nº IV. 67. Cronograma de las primeras actividades



	Nombre de tarea	Duración
33	**3. DESARROLLO**	**67 días**
34	3.1. NOMENCLATURA Y ESTANDARES PARA EL DESARROLLO	3 días
35	**3.2. CAPA DE PRESENTACION**	**20 días**
36	3.2.1. DISEÑO DE INTERFACES DE USUARIO	15 días
37	3.2.2. DISEÑO DE COMPONENTES DE PROCESAMIENTO DE US	5 días
38	**3.3. CAPA DE DATOS**	**39 días**
39	3.3.1. IMPLEMENTACION DE LA BASE DE DATOS	5 días
40	3.3.2. IMPLEMENTACION DE ACCESO A DATOS	12 días
41	3.3.3. DISEÑO DE ESQUEMAS	22 días
42	**3.4. CAPA DE NEGOCIOS**	**4 días**
43	3.4.1. IMPLEMENTACION DE COMPONENTES	4 días
44	**3.5. ESPECIFICACION DE SEGURIDAD**	**7 días**
45	3.5.1. DISEÑO DE ESTRATEGIAS DE AUTORIZACION, AUTEN	7 días
46	**4. ESTABILIZACION**	**69 días**
47	**4.1. REVISION GENERAL DEL SISTEMA**	**54 días**
48	4.1.1. CODIGO FUENTE Y EJECUTABLE	12 días
49	4.1.2. SCRIPTS DE BASE DE DATOS	1 día
50	4.1.3. DOCUMENTACION DE LA INSTALACION	13 días
51	4.1.4. AYUDA	13 días
52	4.1.5. MATERIAL DE CAPACITACION	12 días
53	4.1.6. HISTORIAL DE VERSIONES	3 días
54	4.2. PRUEBAS	15 días
55	**5. INSTALACION**	**5 días**
56	5.1. PLAN DE OPERACIÓN Y SOPORTE	2 días
57	5.2. PROCEDIMINETOS Y PROCESOS	1 día
58	5.3. HISTORIAL DEL SOPORTE TECNICO	2 días

Figura Nº IV. 68. Cronograma de las últimas actividades

En la Figura Nº IV.69, se presenta un resumen cronológico de las actividades principales a ejecutarse para el desarrollo del sistema, indicando además la asignación de recursos del personal en cada una de éstas fases.

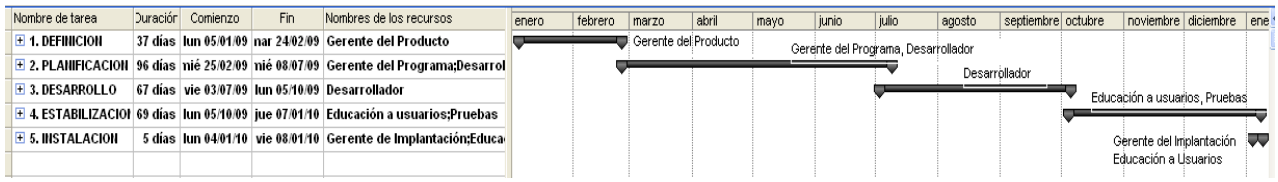


Figura Nº IV. 69. Resumen del cronograma de actividades

En la Figura Nº IV.70, se ilustra el costo del sistema en el que se toma a consideración el tiempo empleado y la asignación de recursos en el cronograma de actividades planteado, en donde el costo de desarrollo de la primera fase de Definición es de $ 110.00, el costo de la segunda fase de Planificación es de $ 210.00, mientras que el costo de la tercera fase de Desarrollo es de $ 130.00; el costo de la cuarta fase de Estabilización es de $ 110.00; en cambio el costo de la última fase de Instalación está en los $ 50,00; dando un total de $ 610.00 por el sistema taxiphone

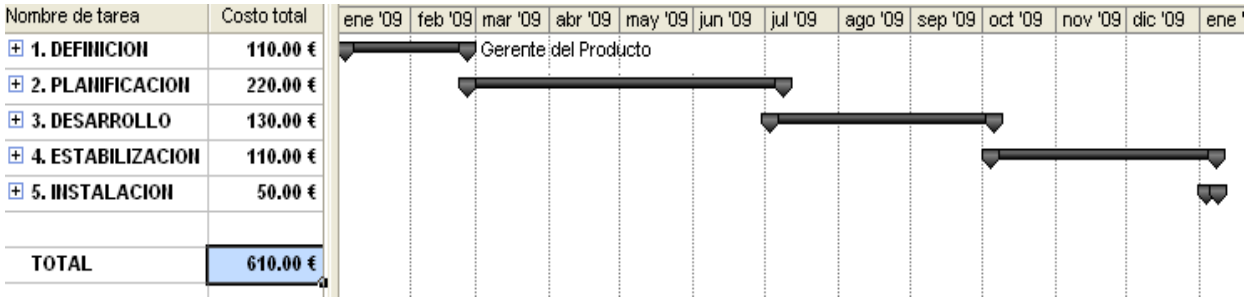


Figura Nº IV. 70. Costo del sistema TAXIPHONE

El costo del sistema está detallado en la tabla Nº IV.6, en la que se plantea costos de desarrollo, gastos de servicios básicos y gastos varios

Tabla Nº IV.6. Costo del sistema Taxiphone

DETALLE	COSTO
Costo de desarrollo	610.00
Servicios Básicos	300.00
Gastos varios	300.00
TOTAL	**1210.00**

4.2. PLANIFICACIÓN

El desarrollo de la planificación está basado en esquematizar o realizar prototipos del sistema a desarrollar basándonos en el ámbito del problema analizado entre todo el equipo de trabajo, llegando a buscar la solución más acertada para conseguir el sistema deseado por el cliente.

4.2.1. Especificación Funcional

Las especificaciones funcionales proveen de la información estudiada y analizada anteriormente, para conseguir resultados que garanticen el correcto desarrollo del sistema tomando en consideración los posibles eventos a presentarse en el manejo del sistema.

4.2.1.1. Diseño Conceptual

Para lograr el diseño conceptual del sistema se debe considerar los primeros aspectos que se consideran como principales ya que a partir de estos apartados se basa la ejecución y desarrollo del sistema en donde se deben cumplir las condiciones, requerimientos, sugerencias y necesidades del cliente.

4.2.1.1.1. Requerimientos

A continuación se detalla las necesidades o requerimientos de la cooperativa para conseguir organización y fluidez en el desenvolvimiento de las actividades de la misma.

Req 1. El sistema permita el ingreso de un nuevo usuario con los datos nombres, dirección, teléfono.

Req 2. Permitir la actualización de la información del usuario dado el nombre del mismo.

Req 3. Poder realizar búsquedas del usuario dado el nombre.

Req 4. Permitir la eliminación o dada de baja del usuario dado el nombre del mismo.

Req 5. Generar listados y reportes de todos los usuarios existentes en la cooperativa

Req 6. Ingresar a un nuevo taxi dado el número, placa, nombre del chofer, modelo, marca, el estado del mismo y alguna observación para el taxi.

Req 7. Realizar la actualización de la información del taxi dado el número del taxi

Req 8. Generar búsquedas de los taxis dado el número del mismo

Req 9. Permitir la eliminación del taxi dado su número.

Req 10. Generar listados y reportes de todos los taxis que existen en la cooperativa.

Req 11. Realizar la actualización del estado de los taxis que se encuentran laborando indicando la hora de inicio de la labor, la hora que va a finalizar, la ubicación en la que se encuentra, la disponibilidad, el destino y la fecha

Req 12. Realizar las actualizaciones de la información de destino, ubicación, o el estado de los taxis que se encuentran laborando dado el número del mismo

Req 13. Realizar la actualización del estado del taxi indicando que se encuentra en descanso

Req 14. Generar listados y reportes de los taxis que han laborado o han descansado dado una fecha

Req 15. Permitir actualizar la disponibilidad del taxi que se encuentra laborando de disponible a ocupado o viceversa.

Req 16. Permitir el registro de una llamada dado el número de teléfono, indicando además el estado si fue o no atendida la misma

Req 17. Generar listados y reportes de todas las llamadas dada la fecha, indicando las que han sido atendidas y las que no

Req 18. Permitir el registro de las carreras realizadas por los taxis con un identificador, la dirección de partida, destino, hora, nombre del usuario y fecha.

Req 19. Generar listados o reportes de todas las carreras atendidas por un taxi dada la fecha

Req 20. Permitir la autenticación de usuarios para el ingreso y manejo del sistema

4.2.1.1.2. Actores

Los actores potenciales del sistema son:

Secretaria

- ✓ **Descripción.-** Persona encargada del manejo de la información dentro de la cooperativa
- ✓ **Ubicación.-** Sede de la cooperativa
- ✓ **Capacitación.-** con sólidos conocimientos en el manejo del sistema operativo XP, pero necesitará capacitación en cuanto al manejo del sistema Taxiphone
- ✓ **Expectativas.-** Contar con u soporte para el manejo de información permitiendo la interacción entre el usuario a través de las llamadas telefónicas y con los taxistas a través del radio enlace, además permita proporcionar reportes de la información requerida.

4.2.1.1.3. Casos de Uso

Los casos de uso están diseñados de acuerdo a los requerimientos planteados al apartado anterior, el fin de los casos de uso es distinguir y aclarecer las actividades que se realizan, actores que intervienen en cada uno de ellos, tomando en consideración además las posibles acciones que se podrían considerar en el sistema.

CU 1. Nuevo usuario

En la Figura Nº IV. 71. Se presenta el primer caso de uso que identifica las actividades que realiza la secretaria para el ingreso de un nuevo usuario

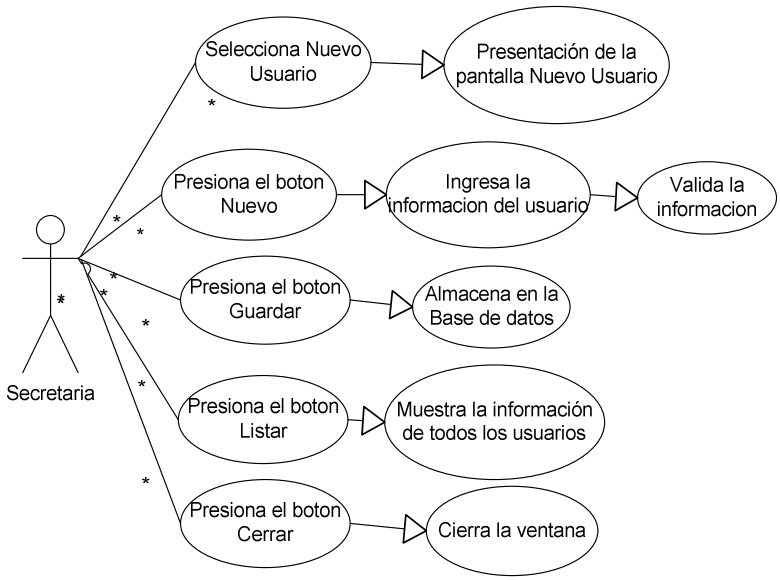


Figura Nº IV. 71. Caso de Uso 1. Nuevo Usuario

CU 2. Modificar Usuario

En la figura Nº IV.72, se muestra el caso de uso para la modificación del usuario

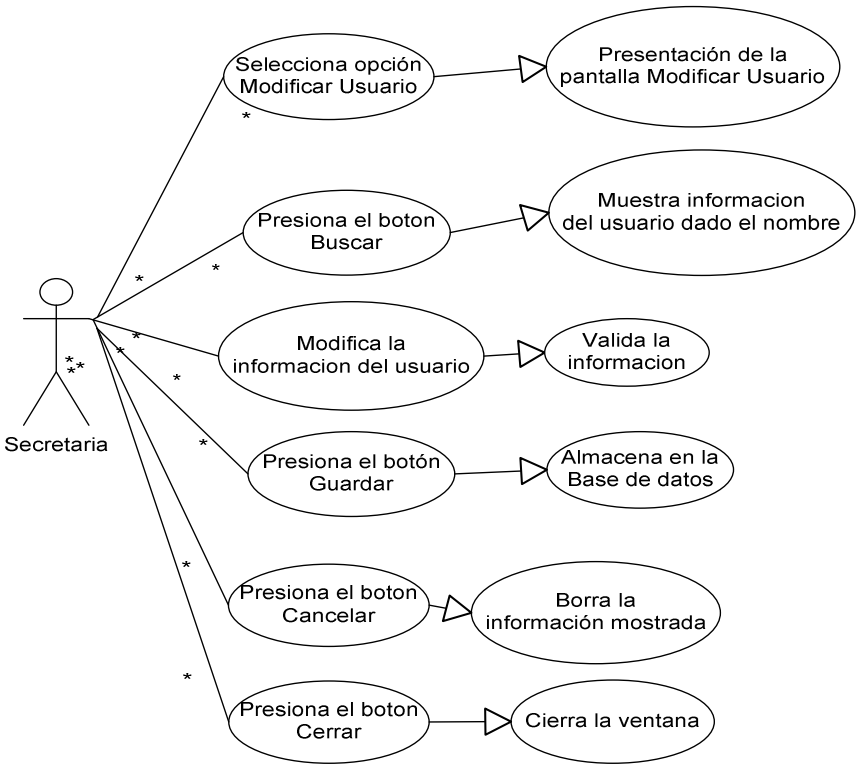


Figura Nº IV. 72. Caso de Uso 2. Actualización del Usuario

CU 3. Eliminar usuario

En la figura Nº IV.73, se presenta el caso de uso para la eliminación de un usuario

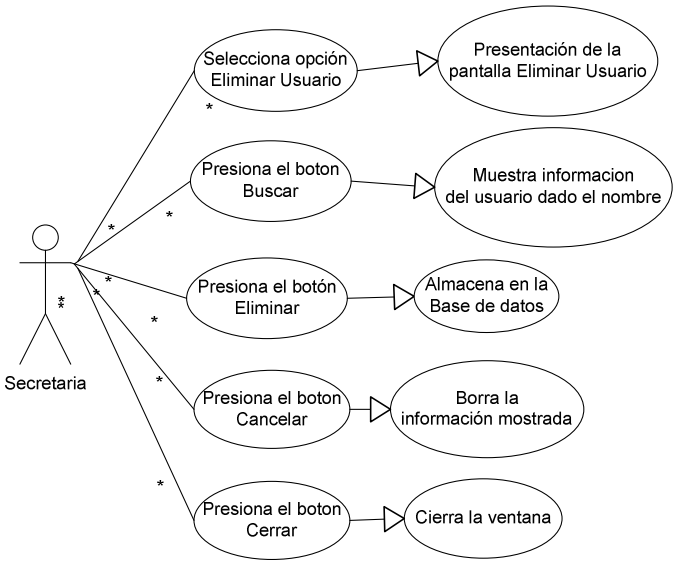


Figura Nº IV. 73. Caso de uso 3. Eliminar Usuario

CU 4. Reportes

A continuación en la Figura Nº IV. 74, se muestra el caso de uso para reportes

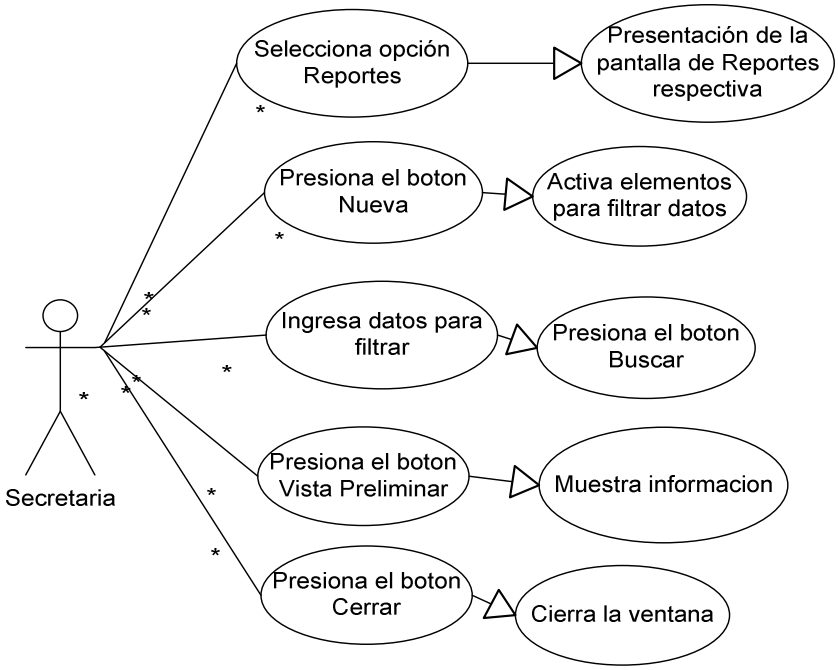


Figura Nº IV. 74. Caso de Uso 4. Reportes

Caso de Uso 5. Nuevo Taxi

En la figura Nº IV.76, se presenta el caso de uso para el ingreso de un nuevo taxi

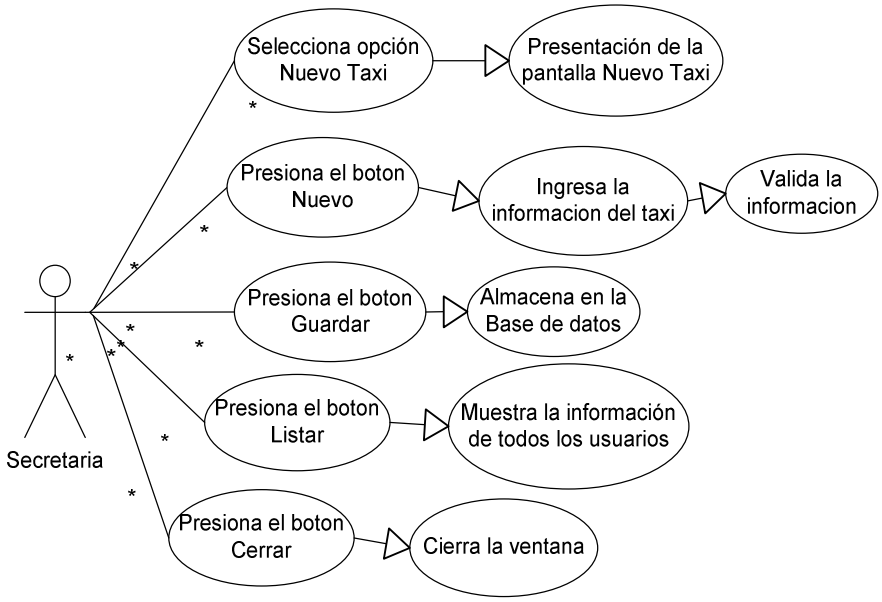


Figura Nº IV. 75. Caso de Uso 5. Nuevo Taxi

Caso de Uso 6. Modificar Taxi

En la figura Nº IV.76, se muestra el caso de uso para modificar la información del taxi

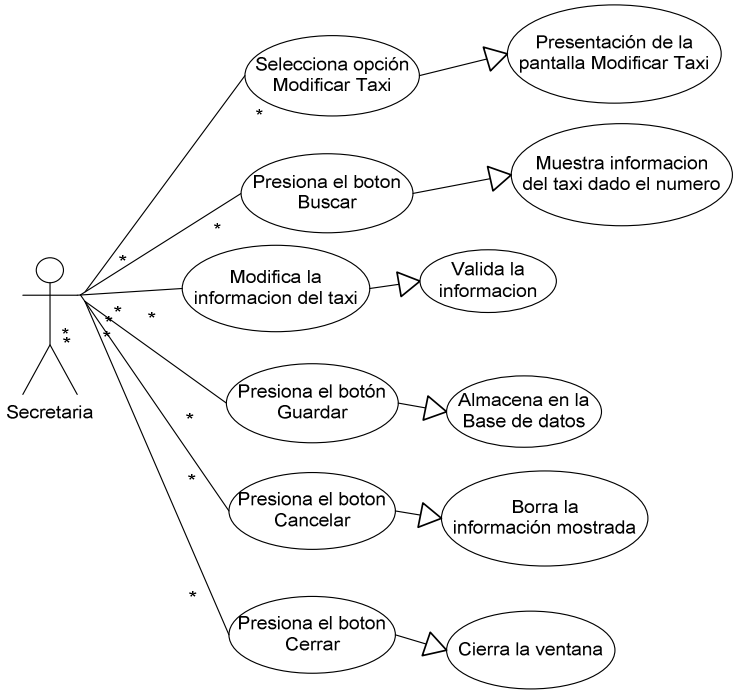


Figura Nº IV. 76. Caso de Uso 6. Modificar Taxi

Caso de Uso 7. Eliminar Taxi

En la figura Nº IV.77. Se presenta el caso de uso para la eliminación de un taxi

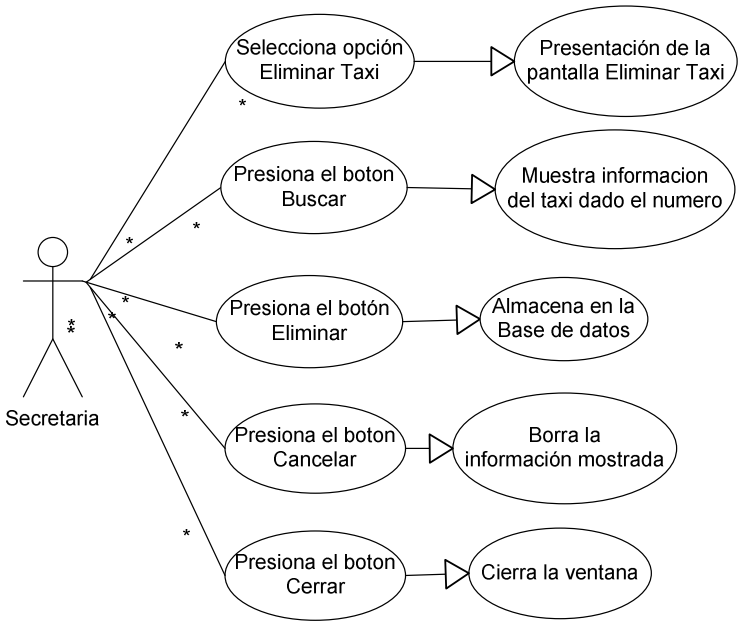


Figura Nº IV. 77. Caso de Uso 7. Eliminar Taxi

Caso de Uso 8. Cambiar Estado

En la figura Nº 78, se muestra el caso de uso para cambiar el estado del taxi

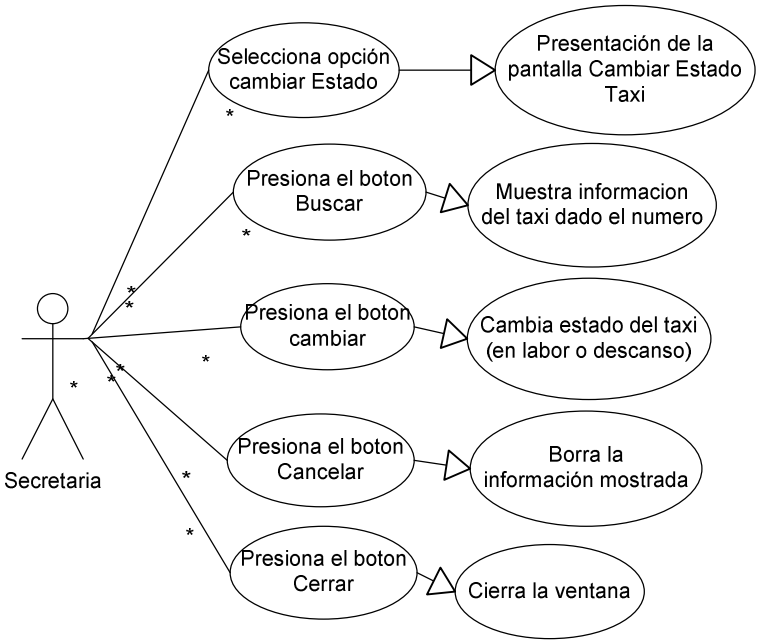


Figura Nº IV. 78. Caso de Uso 8. Cambiar Estado

Caso de Uso 9. Cambiar Disponibilidad

En la figura Nº IV.79, se ilustra el caso de uso para cambiar la disponibilidad de un taxi

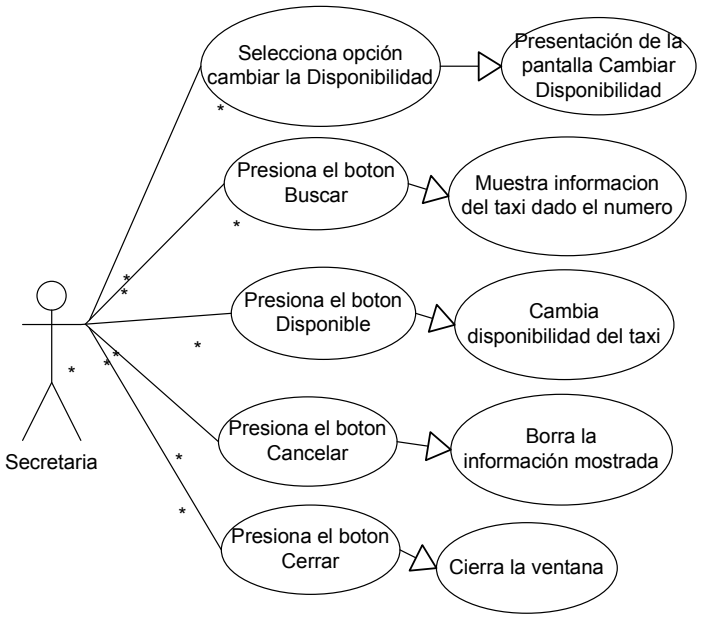


Figura Nº IV. 79. Caso de Uso 9. Cambiar Disponibilidad

Caso de Uso 10. Nueva Llamada

En la siguiente figura Nº IV.80, se ilustra el caso de uso para atender una nueva llamada

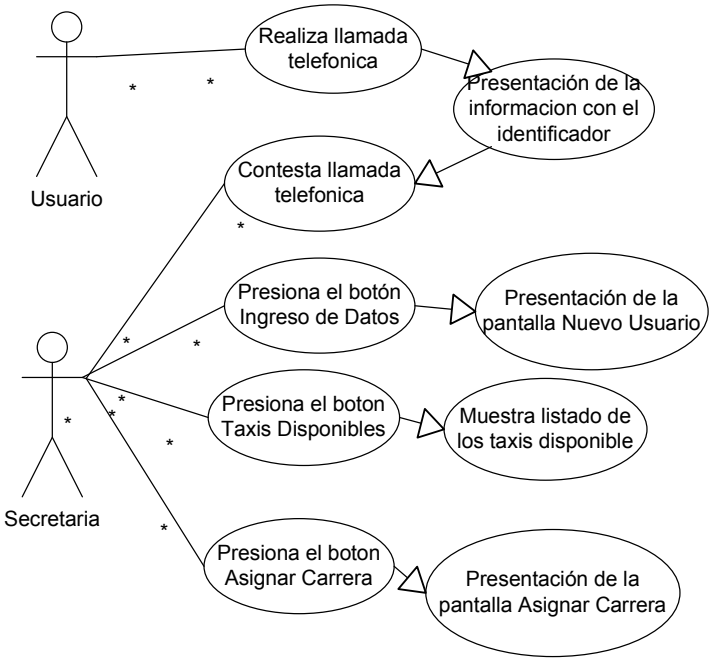


Figura Nº IV. 80. Caso de Uso 10. Nueva Llamada

Caso de Uso 11. Nueva carrera

El la figura Nº IV.81, se presenta el caso de uso para el registro de una nueva carrera

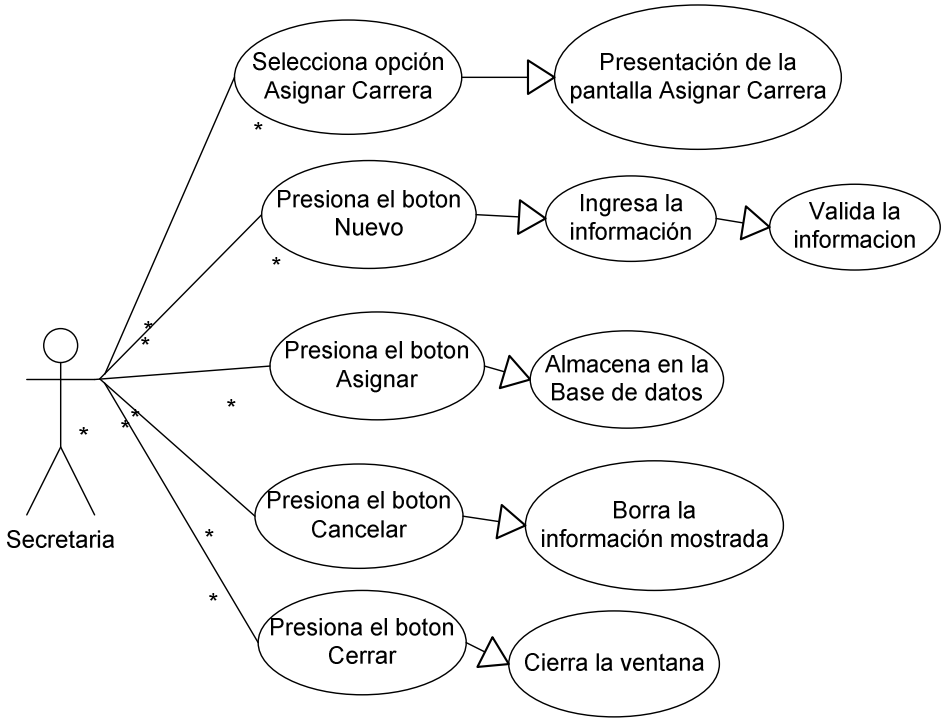


Figura Nº IV. 81. Caso de Uso 11. Nueva carrera

Caso de Uso 12. Autenticación de usuarios

En la figura Nº IV.82, se muestra el caso de uso para la autenticación de usuarios

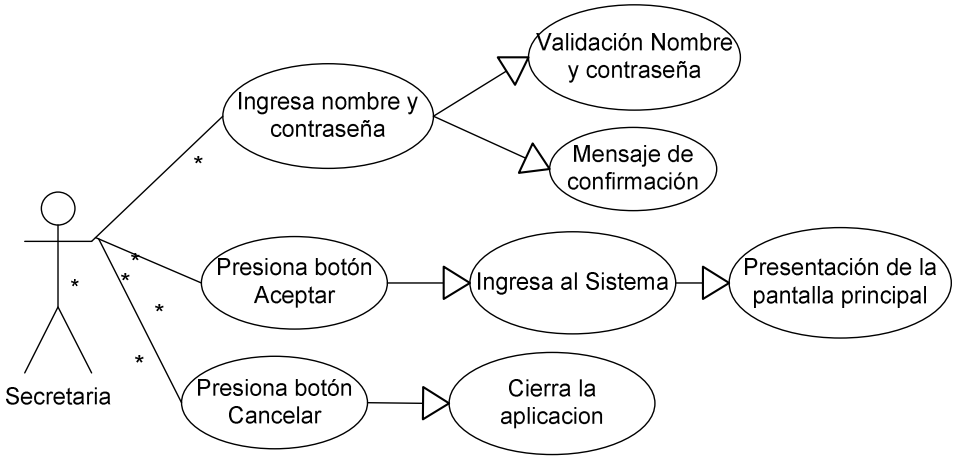


Figura Nº IV. 82. Caso de Uso 12. Autenticación de usuarios

4.2.1.14. Escenarios

A continuación se realiza una breve descripción del proceso de las tareas que se ilustran en los casos de uso presentados en el apartado anterior, en donde se detalla paso a paso cada una de las acciones realizados entre los actores interactuando con el sistema. Las tablas desde el NºIV.7 hasta la tabla Nº IV.18, muestran los escenarios de los casos de uso

Tabla Nº IV.7. Escenario del Caso de Uso 1. Nuevo Usuario

Caso de Uso	CU 1.Nuevo Usuario
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para el ingreso de un nuevo usuario
Visión General	La secretaria selecciona del menú principal la opción Nuevo usuario, en donde se presenta la ventana para el ingreso de datos del nuevo usuario, presiona el botón nuevo para que pueda realizar el ingreso de datos, después pulsa el botón guardar para que se almacenen en la base de datos, se le presenta un mensaje de confirmación, para la verificación de la información podrá listar el botón Listar accediendo a una nueva ventana que podrá presionar el botón listar y se le desplegará la información de todos los usuarios existentes incluido el último usuario ingresado.
CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**

1. La secretaria selecciona del menú principal la opción Nuevo Usuario	2. presenta la ventana respectiva
3.-Presiona el botón Nuevo 4.-Ingresa los datos del usuario	5.-Valida datos
6.- Presiona el botón guardar	7. Almacena en la base de datos
8. Presiona el botón Listar	9. Muestra listado de todos los usuarios
CURSO ALTERNATIVO DE EVENTOS	
5 y 7. Si los datos son incorrectos se presenta un mensaje de error y se regresa al paso 4	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Muestra la información del usuario en el listado de todos los usuarios existentes en la cooperativa

Tabla Nº IV.8. Escenario del Caso de Uso 2. Modificar Usuario

Caso de Uso	CU 2.Modificar Usuario
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para la modificación de la información de un usuario dado su nombre
Visión General	La secretaria selecciona del menú principal la opción modificar usuario, en donde se presenta la ventana para la modificación de datos del usuario, ingresa el nombre del usuario presiona el botón buscar mostrándose la información del usuario, la secretaria puede modificar la información del usuario y presiona el botón guardar para que se

	almacenen en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre
CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Modificar Usuario	2. presenta la ventana respectiva
3.-Ingresa el nombre del usuario a modificar 4.-Presiona el botón Buscar	5.-Muestra la información del usuario
6.-Modifica la información del usuario 7.-Presiona el botón Guardar	8. Almacena en la base de datos
8. Presiona el botón Cerrar	9. Cierra la ventana
CURSO ALTERNATIVO DE EVENTOS	
3 y 6 . Si los datos son incorrectos, presenta mensaje de error y se mantiene en el mismo paso 1. Si no desea guardar los datos modificados puede pulsar el botón Cancelar	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación “Datos modificados correctamente”

Tabla Nº IV.9. Escenario del Caso de Uso 3. Eliminar Usuario

Caso de Uso	CU 3. Eliminar Usuario
Actores	Secretaria

Propósito	Describir el escenario sin problemas del proceso para la eliminación de un o usuario dado su nombre
Visión General	La secretaria selecciona del menú principal la opción eliminar usuario, en donde se presenta la ventana para la eliminación del usuario, ingresa el nombre del usuario presiona el botón buscar mostrándose la información del usuario, la secretaria presiona el botón eliminar para que se actualice la información en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Eliminar Usuario	2. presenta la ventana respectiva
3.-Ingresa el nombre del usuario a eliminar 4.-Presiona el botón Buscar	5.-Muestra la información del usuario
6.-Presiona el botón eliminar	7. Actualiza la base de datos
8. Presiona el botón Cerrar	9. Cierra la ventana

CURSO ALTERNATIVO DE EVENTOS
3. Si los datos son incorrectos se mantiene en el mismo paso 5. Si no desea eliminar al usuario, puede pulsar el botón cancelar

PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación “éxito en la eliminación del usuario”

Tabla Nº IV.10. Escenario del Caso de Uso 4. Reportes

Caso de Uso	CU 4. Reportes
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para la generación de reportes o listados
Visión General	La secretaria selecciona del menú principal la opción para los listados o reportes, en donde se presenta la ventana para la generación del reporte deseado, presiona el botón Nuevo e ingresa los datos para filtrar la información, mostrándose la información necesaria la secretaria pulsa el botón Vista preliminar y se puede presentar la información lista para la impresión

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Reportes	2. presenta la ventana respectiva
3.-Presiona el botón Nuevo 4.-Ingresa el dato para realizar el filtrado de información 5.-Presiona el botón Buscar	6.-Muestra la información del usuario
7.-Presiona el botón Vista Preliminar	8.- Muestra información en reporte
9. Presiona el botón Cerrar	10. Cierra la ventana
CURSO ALTERNATIVO DE EVENTOS	
4. Si los datos son incorrecto se mantiene en el mismo paso	

PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Información deseada

Tabla Nº IV.11. Escenario del Caso de Uso 5. Nuevo Taxi

Caso de Uso	CU 5. Nuevo Taxi
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para el ingreso de un nuevo taxi
Visión General	La secretaria selecciona del menú principal la opción Nuevo Taxi, en donde se presenta la ventana para el ingreso de datos del nuevo taxi, presiona el botón nuevo para que pueda realizar el ingreso de datos, después pulsa el botón guardar para que se almacenen en la base de datos, se le presenta un mensaje de confirmación, para la verificación de la información podrá listar el botón Listar accediendo a una nueva ventana que podrá presionar el botón listar y se le desplegará la información de todos los taxis existentes incluido el último taxi ingresado.

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Nuevo Taxi	2. presenta la ventana respectiva
3,.Presiona el botón Nuevo	

4.-Ingresa los datos del taxi	5.-Valida datos
6. Presiona el botón guardar	7. Almacena en la base de datos
8. Presiona el botón Listar	9. Muestra listado de todos los taxis
CURSO ALTERNATIVO DE EVENTOS	
4. Si los datos son incorrectos se presenta un mensaje de error	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Muestra la información del usuario en el listado de todos los taxis existentes en la cooperativa

Tabla Nº IV.12. Escenario del Caso de Uso 6. Modificar taxi

Caso de Uso	CU 6.Modificar taxi
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para la modificación de la información de un taxi dado su número
Visión General	La secretaria selecciona del menú principal la opción modificar taxi, en donde se presenta la ventana para la modificación de datos del taxi, ingresa el número del taxi presiona el botón buscar mostrándose la información del taxi, la secretaria puede modificar la información del taxi y presiona el botón guardar para que se almacenen en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre
CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**

1. La secretaria selecciona del menú principal la opción Modificar Taxi	2. presenta la ventana respectiva
3.-Ingresa el número del taxi a modificar 4.-Presiona el botón Buscar	5.-Muestra la información del taxi
6.-Modifica la información del taxi 7.-Presiona el botón Guardar	8. Almacena en la base de datos
9. Presiona el botón Cerrar	10. Cierra la ventana
CURSO ALTERNATIVO DE EVENTOS	
3 y 6 . Si los datos son incorrectos, presenta mensaje de error y se mantiene en el mismo paso 7.- Si no desea guardar los datos modificados puede pulsar el botón Cancelar	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación "Datos modificados correctamente"

Tabla Nº IV.13. Escenario del Caso de Uso 7. Eliminar Taxi

Caso de Uso	CU 7. Eliminar Taxi
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para la eliminación de un taxi dado su número
Visión General	La secretaria selecciona del menú principal la opción eliminar taxi, en donde se presenta la ventana para la eliminación del taxi, ingresa el

	número del taxi presiona el botón buscar mostrándose la información del taxi, la secretaria presiona el botón eliminar para que se actualice la información en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Eliminar taxi	2. presenta la ventana respectiva
3.-Ingresa el número del taxi a eliminar 4.-Presiona el botón Buscar	5.-Muestra la información del taxi
6.-Presiona el botón eliminar	7. Actualiza la base de datos
8. Presiona el botón Cerrar	9. Cierra la ventana

CURSO ALTERNATIVO DE EVENTOS	
3. Si los datos son incorrectos se mantiene en el mismo paso 6. Si no desea eliminar al taxi, puede pulsar el botón cancelar	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación “éxito en la eliminación del taxi”

Tabla Nº IV.14. Escenario del Caso de Uso 8. Cambiar Estado

Caso de Uso	CU 8. Cambiar Estado
Actores	Secretaria

Propósito	Describir el escenario sin problemas del proceso para el cambio de estado del taxi de taxi en descanso a taxi en labor o viceversa
Visión General	La secretaria selecciona del menú principal la opción Cambiar Estado, en donde se presenta la ventana para la cambiar el estado del taxi, ingresa el número del taxi presiona el botón buscar mostrándose la información del taxi, la secretaria presiona el botón cambiar estado y presiona el botón Cambiar para que se almacene en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción cambiar estado	2. presenta la ventana respectiva
3.-Ingresa el número del taxi a cambiar 4.-Presiona el botón Buscar	5.-Muestra la información del taxi
6.-Cambia el estado del taxi 7.-Presiona el botón Cambiar	8. Almacena en la base de datos
9. Presiona el botón Cerrar	10. Cierra la ventana
CURSO ALTERNATIVO DE EVENTOS	
3 . Si los datos son incorrectos se mantiene en el mismo paso	

PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación “éxito en el cambio de estado del taxi”

Tabla Nº IV.15. Escenario del Caso de Uso 9. Cambiar Disponibilidad

Caso de Uso	CU 9. Cambiar Disponibilidad
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para el cambio de disponibilidad del taxi que se encuentra en labor de disponible a ocupado o viceversa
Visión General	La secretaria selecciona del menú principal la opción Cambiar Disponibilidad, en donde se presenta la ventana para la cambiar la disponibilidad del taxi, ingresa el número del taxi presiona el botón buscar mostrándose la información del taxi, la secretaria presiona el botón cambiar disponibilidad y presiona el botón Cambiar para que se almacene en la base de datos, la secretaria pulsa el botón cerrar para que la ventana se cierre

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción cambiar disponibilidad	2. presenta la ventana respectiva
3.-Ingresa el número del taxi a cambiar 4.-Presiona el botón Buscar	5.-Muestra la información del taxi
6.-Cambia la disponibilidad del taxi 7.-Presiona el botón Cambiar	8. Almacena en la base de datos
9. Presiona el botón Cerrar	10. Cierra la ventana

CURSO ALTERNATIVO DE EVENTOS	
3 . Si los datos son incorrectos se mantiene en el mismo paso	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Mensaje de confirmación "éxito en el cambio de disponibilidad"

Tabla Nº IV.16. Escenario del Caso de Uso 10. Nueva Llamada

Caso de Uso	CU 10. Nueva Llamada
Actores	Secretaria, Usuario
Propósito	Describir el escenario sin problemas del proceso para la atención de una nueva llamada telefónica por parte del usuario
Visión General	La secretaria selecciona del menú principal la opción atender Llamada en donde se presenta la ventana respectiva con la información de la llamada telefónica, presiona el botón Ingreso de datos en caso no se encuentren registrados los datos del usuario, la secretaria ingresa los datos, luego pulsa el botón taxis disponibles y asigna un taxi en labor pulsando el botón Asignar.

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria selecciona del menú principal la opción Atender llamada	2. presenta la ventana respectiva
3. Presiona el botón Atender llamada	4. Muestra los datos de la llamada

5 Presiona el botón nuevo Usuario 7. Ingresa la información del usuario	6.. presenta la ventana respectiva 8.Valida datos
9. Presiona el botón Taxis disponibles	10. Muestra los taxis disponibles
11. Presiona el botón asigna	12. Actualiza la información en la base de datos
CURSO ALTERNATIVO DE EVENTOS	
8. Si los datos son incorrectos se presenta un mensaje de error y se regresa al paso 7	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Atención a la llamada telefónica por parte del usuario

Tabla Nº IV.17. Escenario del Caso de Uso 11. Nuevo Carrera

Caso de Uso	CU 11. Nueva Carrera
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para el ingreso de una nueva carrera
Visión General	La secretaria selecciona del menú principal la opción Asignar carrera, en donde se presenta la ventana para el ingreso de datos de la nueva carrera, presiona el botón nuevo para que pueda realizar el ingreso de datos, después pulsa el botón guardar para que se almacenen en la base de datos, se le presenta un mensaje de confirmación, la secretaria pulsa el botón cerrar y se cierra la ventana
CURSO TÍPICO DE EVENTOS	

Acción del Actor	Acción del Actor
1. La secretaria selecciona del menú principal la opción Asignar carrera	2. presenta la ventana respectiva
3.-Presiona el botón Nuevo 4.- Ingresa los datos del taxi	5.-Valida datos
6. Presiona el botón guardar	7. Almacena en la base de datos
8. Presiona el botón Cerrar	9. Cierra la ventana
CURSO ALTERNATIVO DE EVENTOS	
5. Si los datos son incorrectos regresa al paso 4	
PRE-CONDICIÓN	Para acceder al menú principal se debe autenticar como usuario del sistema
POST-CONDICIÓN	Asignar taxi a la carrera solicitada por el usuario

Tabla Nº IV.18. Escenario del Caso de Uso 12. Autenticación de Usuarios

Caso de Uso	CU 12. Autenticación de Usuarios
Actores	Secretaria
Propósito	Describir el escenario sin problemas del proceso para la autenticación de usuarios
Visión General	La secretaria accede a la aplicación en donde se presenta una ventana de autenticación, ingresando e nombre y contraseña, si los datos son correctos se podrá ingresar a la pantalla principal con su menú de opciones

CURSO TÍPICO DE EVENTOS	
Acción del Actor	**Respuesta del Sistema**
1. La secretaria accede al sistema	2. Presenta la ventana de autenticación
3.-Ingresa nombre y contraseña 4.-Presiona el botón Aceptar	5.Valida datos y presenta mensaje de confirmación 6. Muestra pantalla con menú principal
CURSO ALTERNATIVO DE EVENTOS	
5. Si los datos son incorrectos regresa al paso3	
PRE-CONDICIÓN	Acceder al sistema teniendo la pantalla de autenticación inicial
POST-CONDICIÓN	Pantalla principal con el menú de opciones

2.1.1.5.Glosario de Términos

En la tabla Nº IV.19, se presenta el glosario de términos que se emplea en todo el proceso de desarrollo de la metodología Microsoft Solution Framework MSF.

Tabla Nº IV.19. Glosario de Términos

TERMINO	DESCRIPCIÓN
Ámbito	Entorno en el que se va a desarrollar la aplicación software.
Base de datos	Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso.
Calidad	Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.

Carrera	Acción realizada por los taxis de un punto a otro
Codificación	Consiste en proporcionar códigos numéricos o alfanuméricos a diversos procesos para llevar un seguimiento y control más profundo de alguna actividad
Contraseña o password	Es un código o una palabra que se utiliza para acceder a datos restringidos de un ordenador.
Cooperativa de Taxis	Es una organización donde realiza el servicio de transportar a usuarios de un destino a otro
Desarrollador	Es un programador que se dedica a una o más facetas del proceso de desarrollo de software, un ámbito algo más amplio de la programación
Desarrollo	Proceso de transformación de un diseño en componentes hardware y/o software.
Diseño	Proceso de definición de la arquitectura, componentes, interfaces, y otras
Hardware	Dispositivo físico que ayudan a la elaboración de un sistema.
Implementación	Llevar a cabo la creación de un proceso que fue previamente determinado.
Interfaz	Ambiente de trabajo del usuario.
Llamada	Acción de solicitar el servicio de taxis
Procesos	Conjunto de actividades interrelacionadas que usan recursos para transformar entradas en salidas.
Producto	Resultado de un proceso. Un producto software esta relacionado con cuatro categorías de productos como son el hardware, software,

	comunicaciones, servicios.
Proyecto	Conjunto de actividades planificadas y coordinadas, controladas, presupuestadas, y documentadas con fechas de comienzo y finalización, que se emprende para alcanzar unos objetivos conforme a requisitos específicos, por una organización temporal adaptada a sus necesidades.
Requisito	Necesidad o expectativa que se establece de forma explícita o implícita.
Secretaria	Persona encargada del manejo del sistema Taxiphone
Sistema	Conjunto de elementos interrelacionados e interactuantes en uno o más de los procesos que proporcionan la capacidad de satisfacer una necesidad u objetivo definido.
Sistema Operativo	Conjunto de programas destinados a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera eficiente
Software	Conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.
Taxi	Medio de transporte que brinda el servicio de movilización de un punto a otro
Usuario	Cliente que requiere el servicio de la cooperativa de taxis
Usuario del sistema	Una persona que usa el sistema para realizar una función específica.

Validación	Confirmación mediante examen y provisión de evidencia objetiva de que se cumplen los requisitos particulares para ser usado con un propósito específico y que satisface las necesidades del cliente.
Visión	Permite dar claridad sobre lo que se quiere decir y adónde se desea llegar
Visual .Net	Lenguaje de programación en el que se va a codificar el sistema propuesto.

4.2.1.2. Diseño Lógico

En el diseño lógico se describirá la solución en términos de su organización, su estructura y la interacción de sus partes desde la perspectiva del desarrollador, buscando generalizar la estructura técnica del sistema a través de los diagramas de secuencia, clase, interfaces de usuario, que permitirán obtener prototipos del sistema a realizar.

4.2.1.2.1. Tecnología a utilizar en el proyecto

Para el desarrollo del sistema Taxiphone, se lo realizará en el lenguaje C# bajo la plataforma .NET 2005 empleando el framework NHibernate, para comprobar su funcionalidad y proceso de desarrollo; la base de datos reposará en el motor SQL Server 2005 con el que interactuará directamente

4.2.1.2.2. Diagramas de secuencia

Las figuras Nº IV.83 hasta la figura Nº IV.94, se muestran los diagramas de secuencia que se presentan en sistema Taxiphone

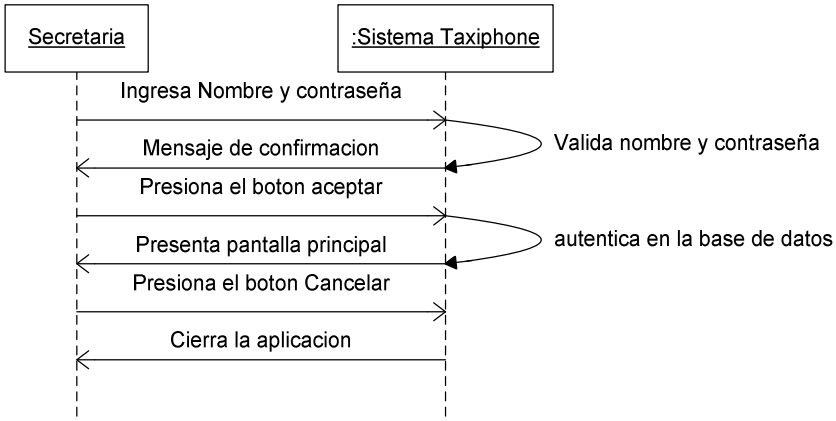


Figura Nº IV. 83. Diagrama de Secuencia: Autenticación de Usuarios

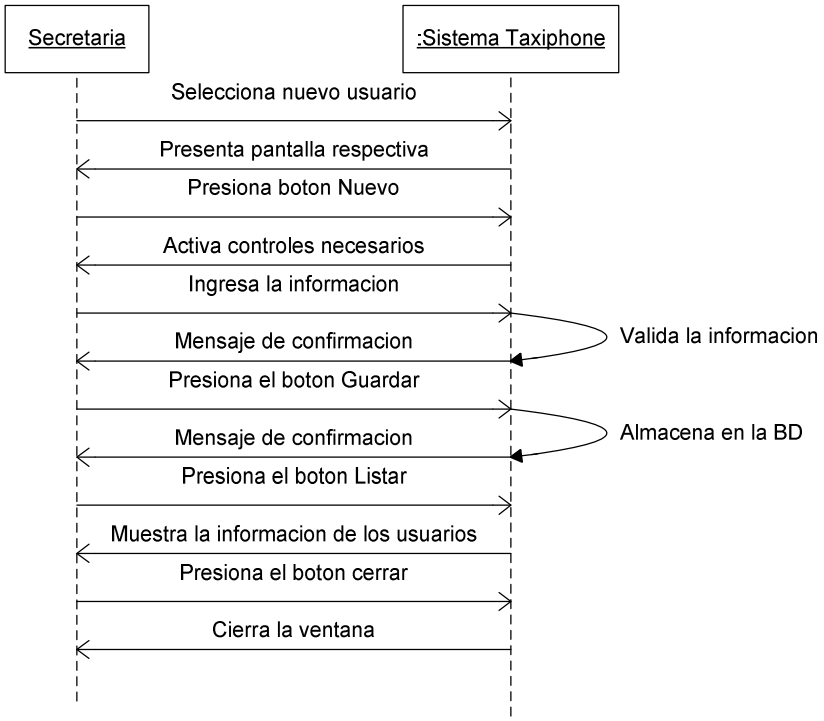


Figura Nº IV. 84. Diagrama de Secuencia. Nuevo Usuario

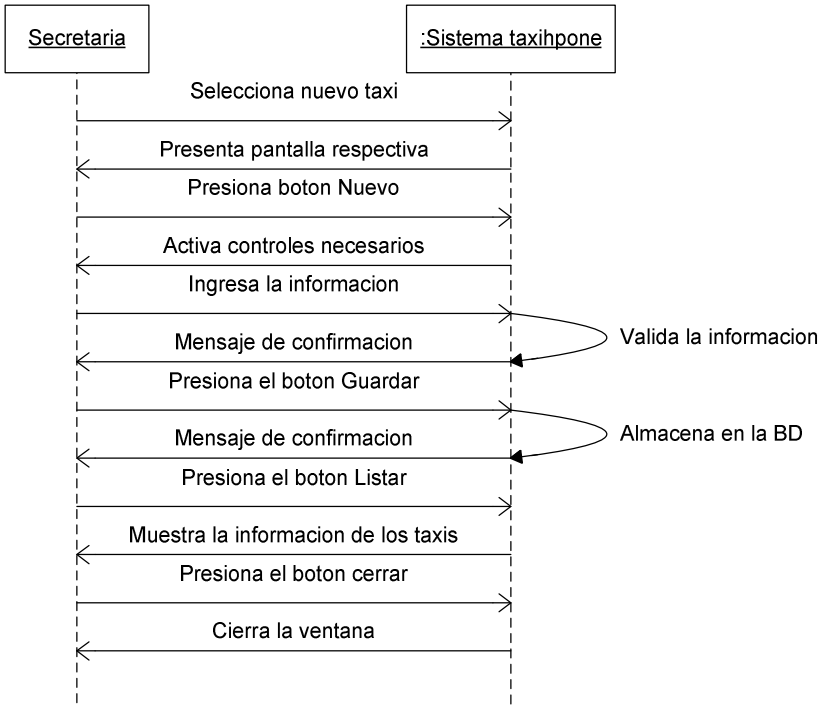


Figura Nº IV. 85. Diagrama de Secuencia: Nuevo Taxi

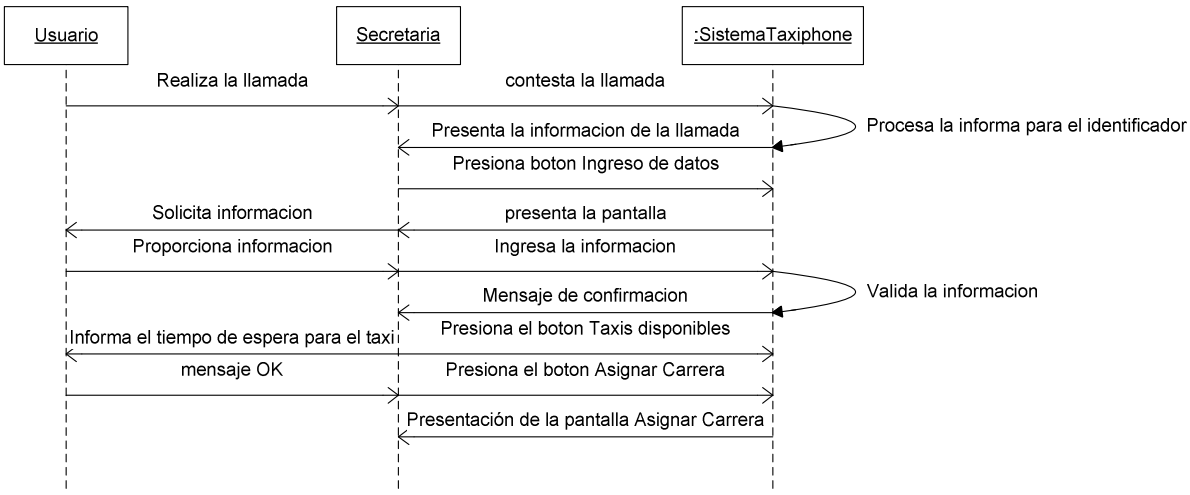


Figura Nº IV. 86. Diagrama de Secuencia: Llamada entrante

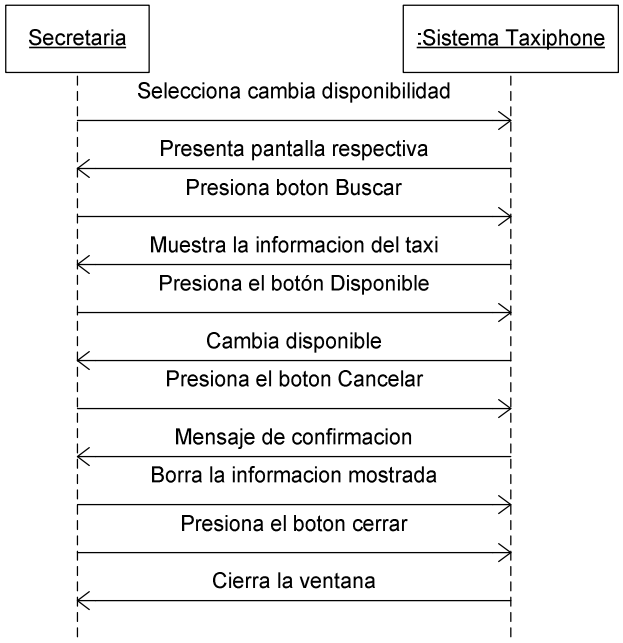


Figura Nº IV. 87. Diagrama de Secuencia: Cambiar Disponibilidad

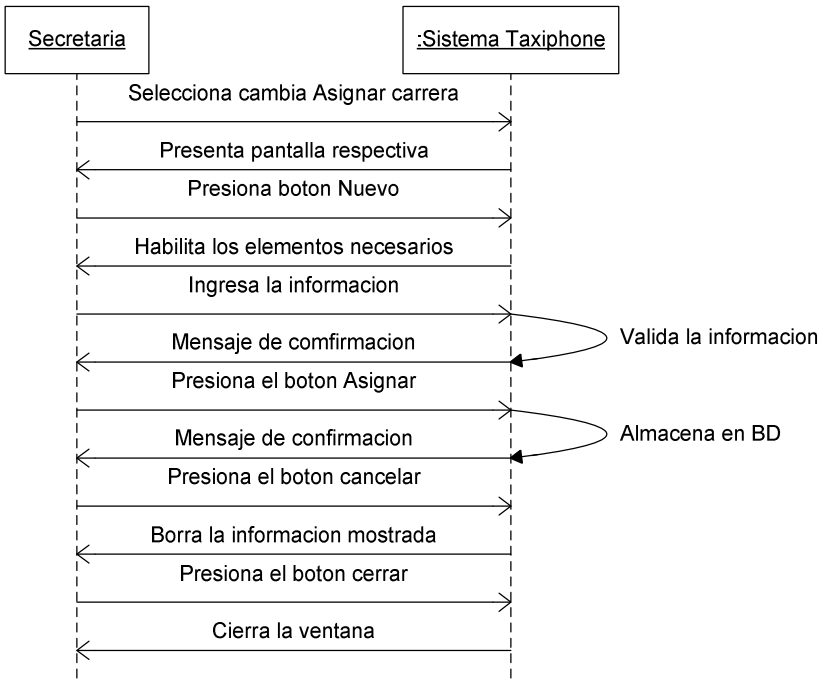


Figura Nº IV. 88. Diagrama de Secuencia: Asignar Carrera

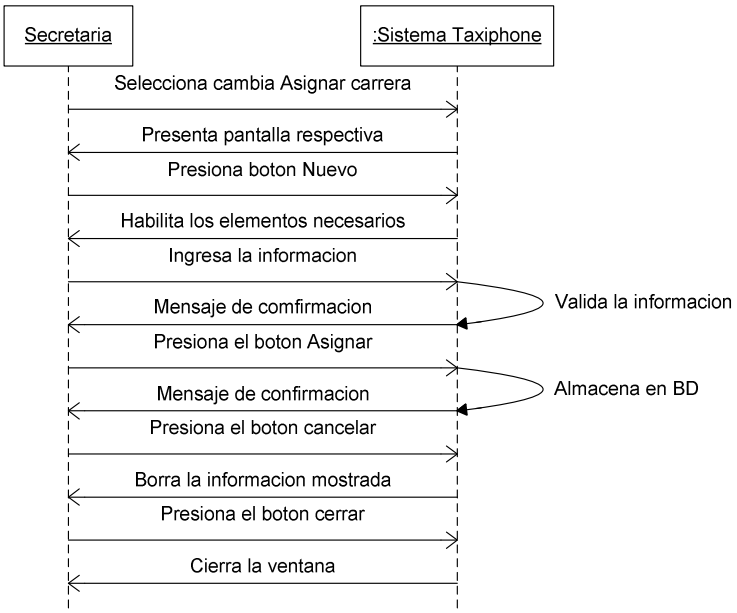


Figura Nº IV. 89. Diagrama de Secuencia: Cambiar Estado

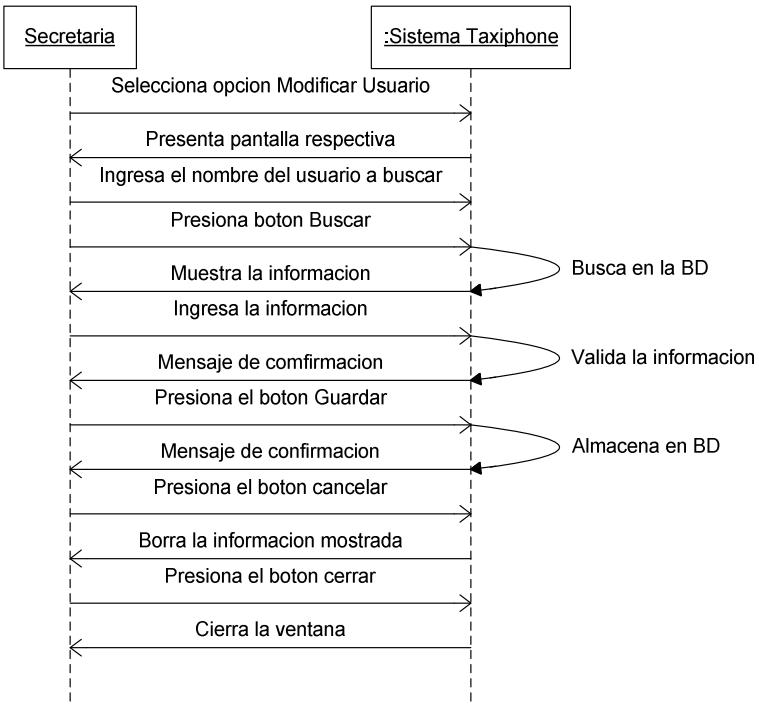


Figura Nº IV. 90. Diagrama de Secuencia: Modificar usuario

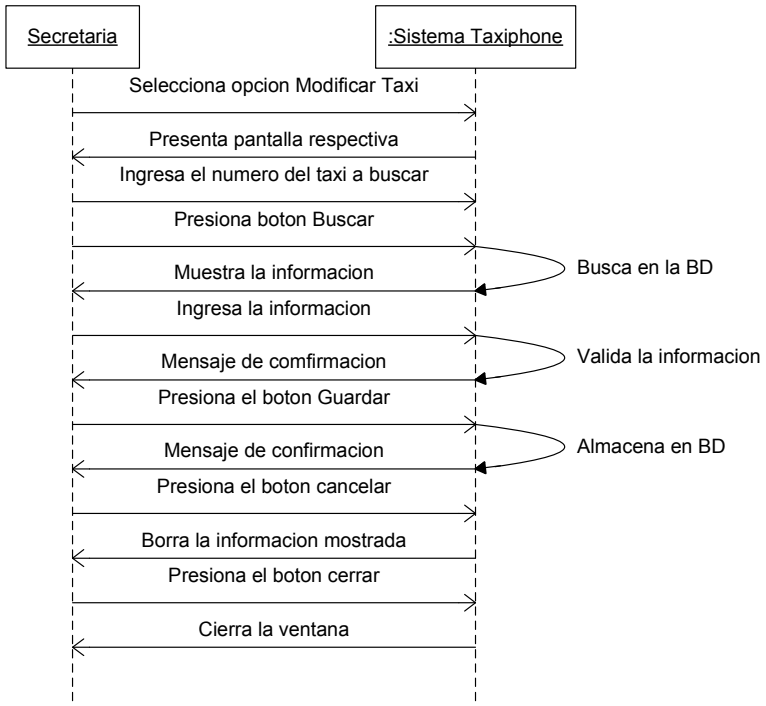


Figura Nº IV. 91. Diagrama de Secuencia: Modificar Taxi

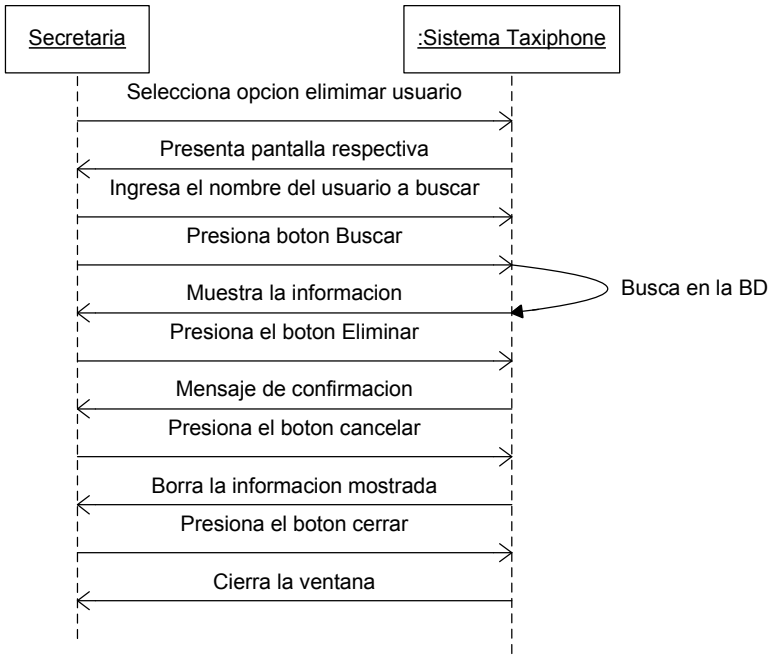


Figura Nº IV. 92. Diagrama de Secuencia: Eliminar Usuario

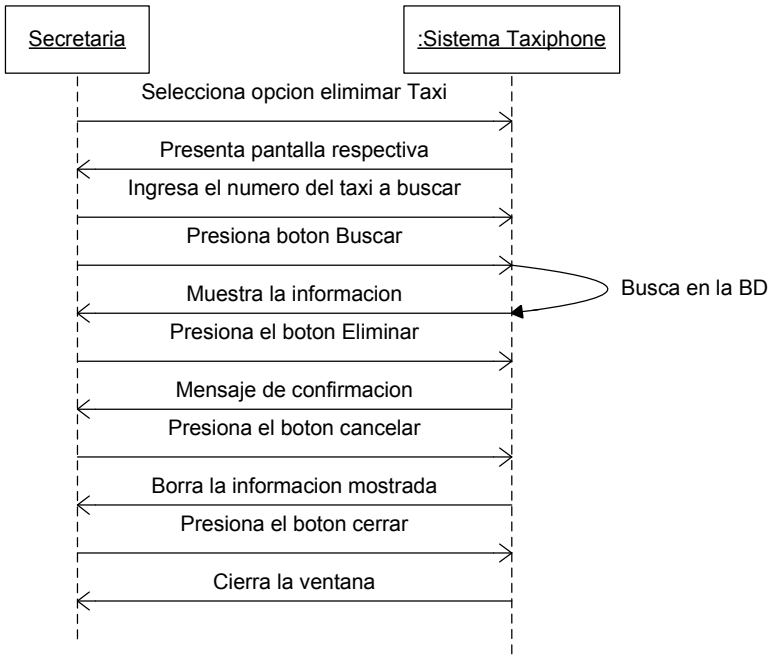


Figura Nº IV. 93. Diagrama de Secuencia: Eliminar Taxi

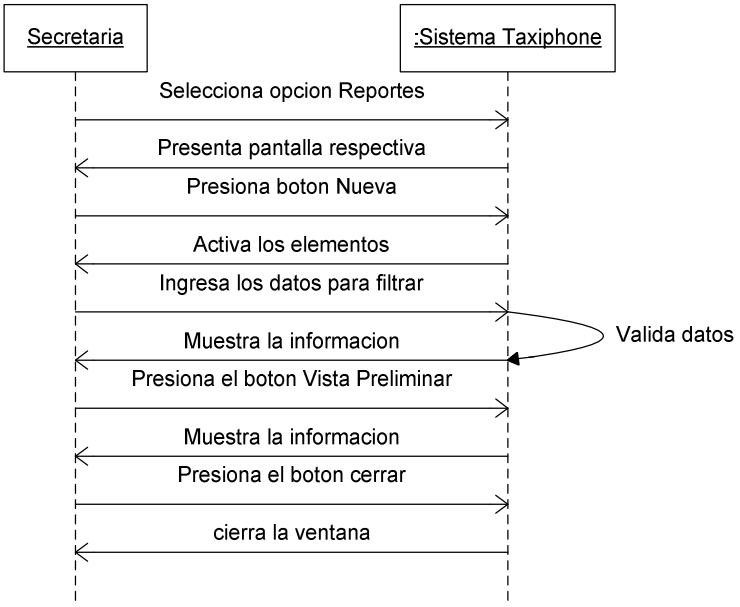


Figura Nº IV. 94. Diagrama de Secuencia: Reportes

4.2.1.2.3. Diagramas de Clases

La figura Nº IV. 95, se muestra el diagrama de clases en el modelo lógico del sistema Taxiphone, en el que se presenta las posibles clases de la aplicación y la relación que se presenta entre ellas

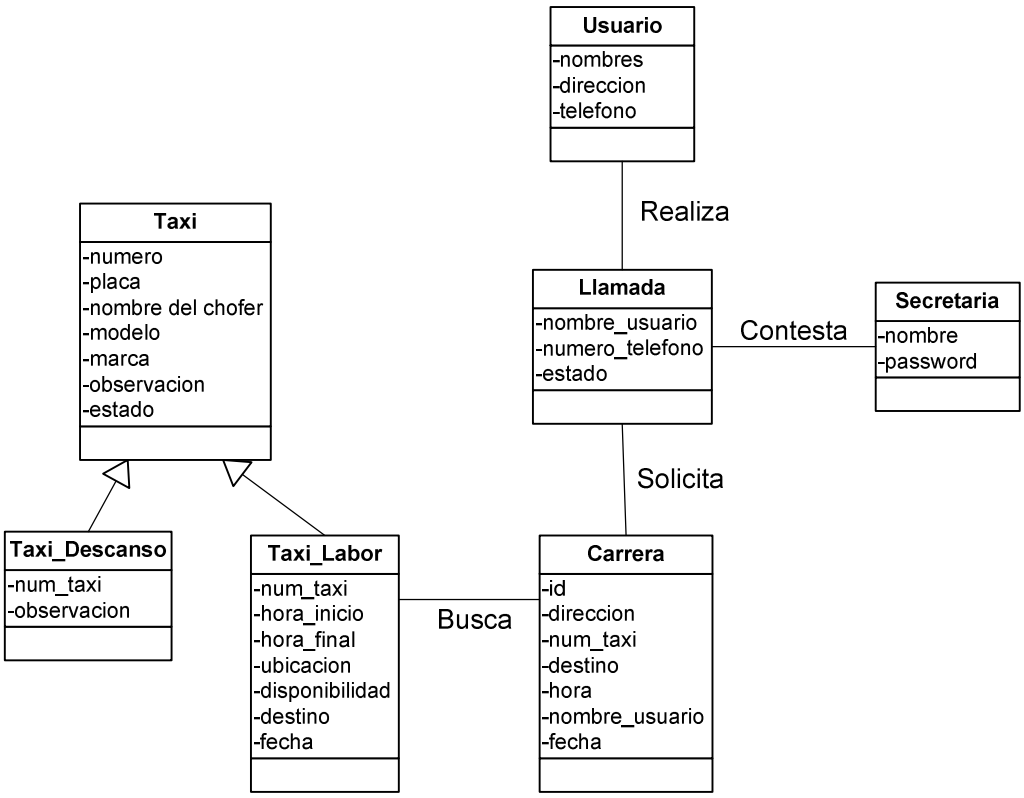


Figura Nº IV. 95. Diagrama de clases

4.2.1.2.4. Modelo Lógico de datos

En la figura Nº IV.96, se presenta el modelo lógico de datos, representados por un diagrama entidad relación semejante a la que se encontrará constituida en la base de datos, con sus tablas, campos y relaciones

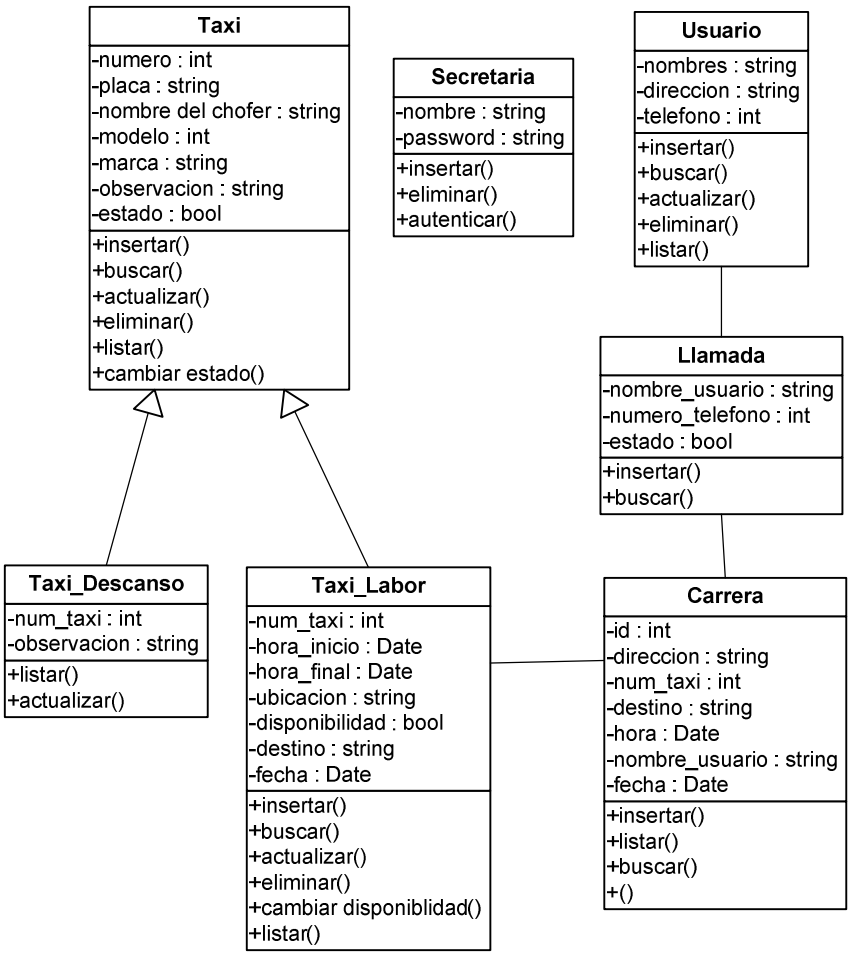


Figura Nº IV. 96. Modelo lógico de datos

4.2.1.2.5. Diseño de interfaces de Usuario

A continuación se presenta un bosquejo de las posibles interfaces del sistema taxiphone, las mismas que podrían variar de acuerdo a las necesidades de funcionalidad o de diseño, las mismas que se encuentran contempladas a partir de la figura Nº IV.97 hasta la figura Nº IV.104

Figura Nº IV. 97. Pantalla de autenticación

Figura Nº IV. 98. Pantalla para insertar usuario

Figura Nº IV. 99. Pantalla para cambiar el estado del taxi

Figura Nº IV. 100. Pantalla para asignar carrera

Figura Nº IV. 101. Pantalla para cambiar la disponibilidad

Figura Nº IV. 102. Pantalla para contestar una llamada entrante

Figura Nº IV. 103. Pantalla para modificar usuario

Figura Nº IV. 104. Pantalla para listar usuarios

4.2.1.3. Diseño Físico

El paso de un modelo lógico a uno físico requiere un profundo entendimiento del manejador de bases de datos que se desea emplear, en el que se debe contemplar, las entidades, atributos, tipos de datos, relaciones, claves primarias etc, es decir, el modelo físico, estará enfocado más al desarrollo, en el que se involucra el proceso de desarrollo e la aplicación.

4.2.1.3.1. Diagramas de secuencia refinados

De acuerdo a los diagramas de secuencia presentados en el modelo lógico se procedes a refinarlos, en el que se indica los campos que intervienen en cada proceso, estos diagramas se encuentran representados en las figuras Nº IV.105 hasta la figura Nº IV.116, presentadas a continuación.

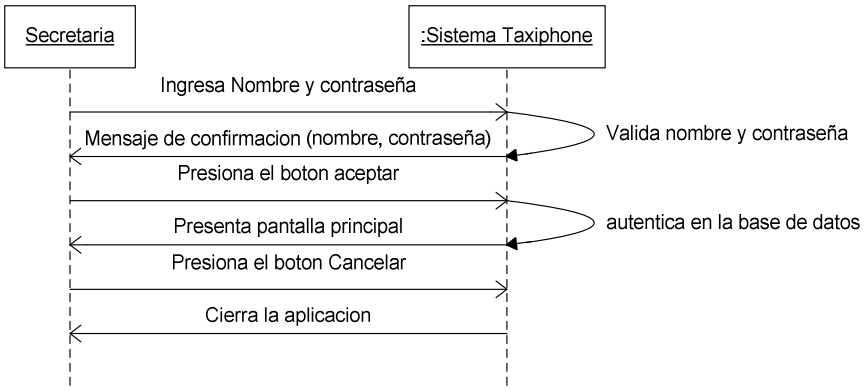


Figura Nº IV. 105. Diagrama de secuencia Refinado: Autenticación de usuarios

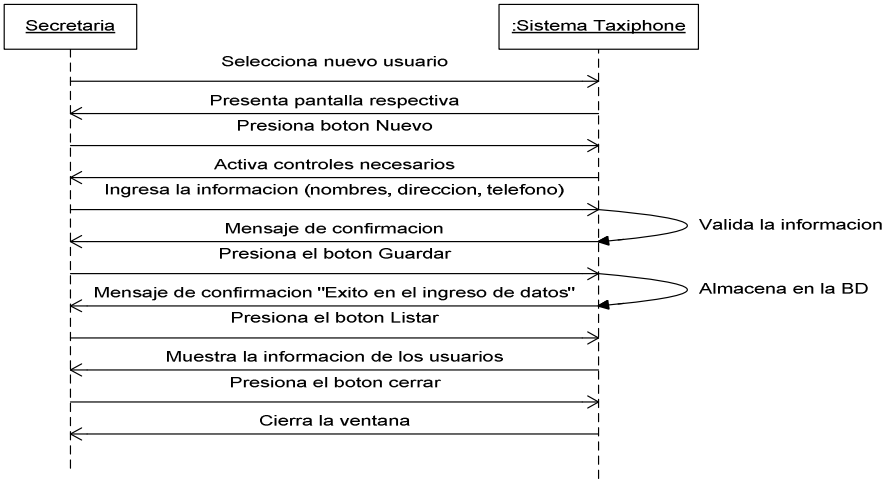


Figura Nº IV. 106. Diagrama de secuencia refinado: Nuevo Usuario

Figura Nº IV. 107. Diagrama de secuencia refinado: Nuevo taxi

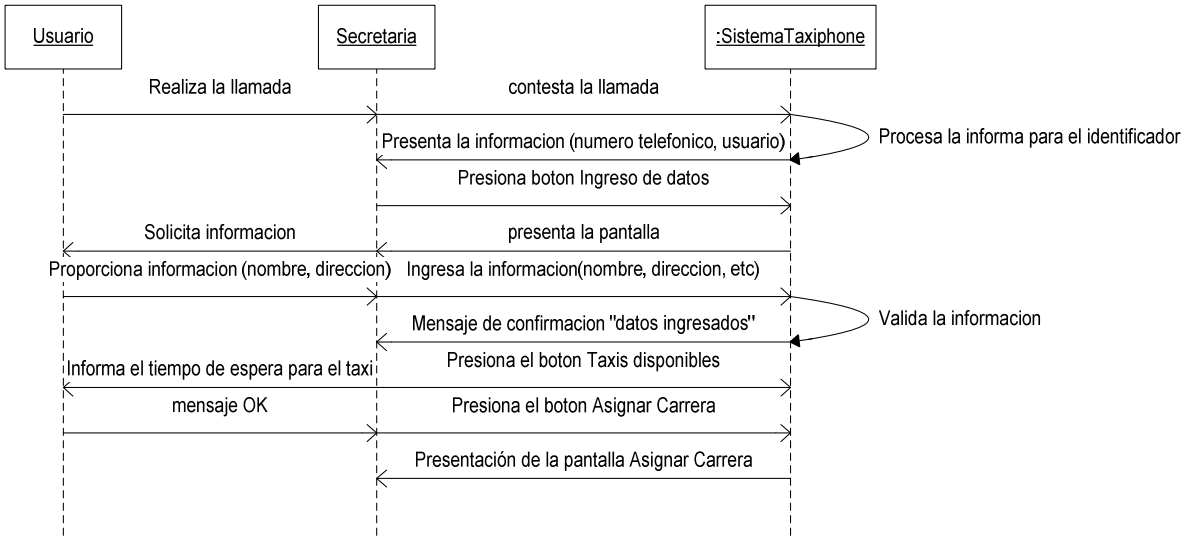


Figura Nº IV. 108. Diagrama de secuencia refinado. Llamada entrante

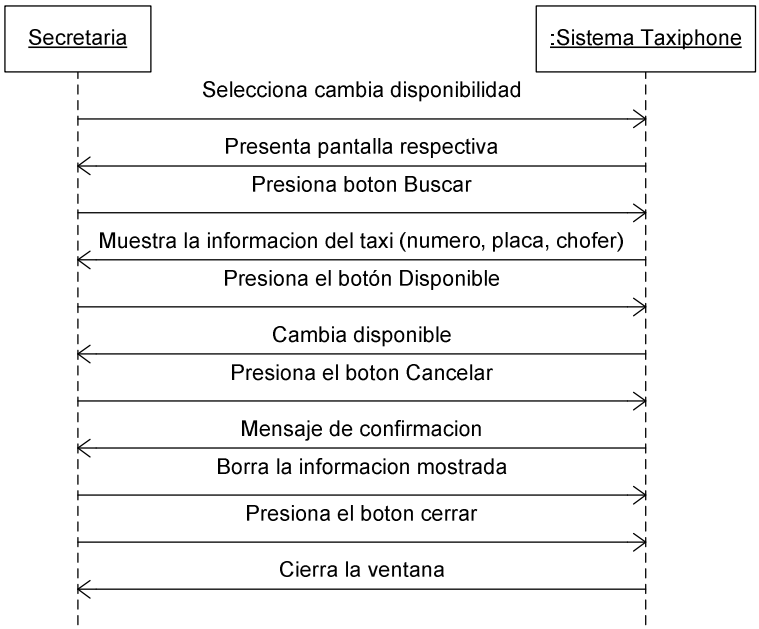


Figura Nº IV. 109. Diagrama de secuencia refinado: Cambiar disponibilidad

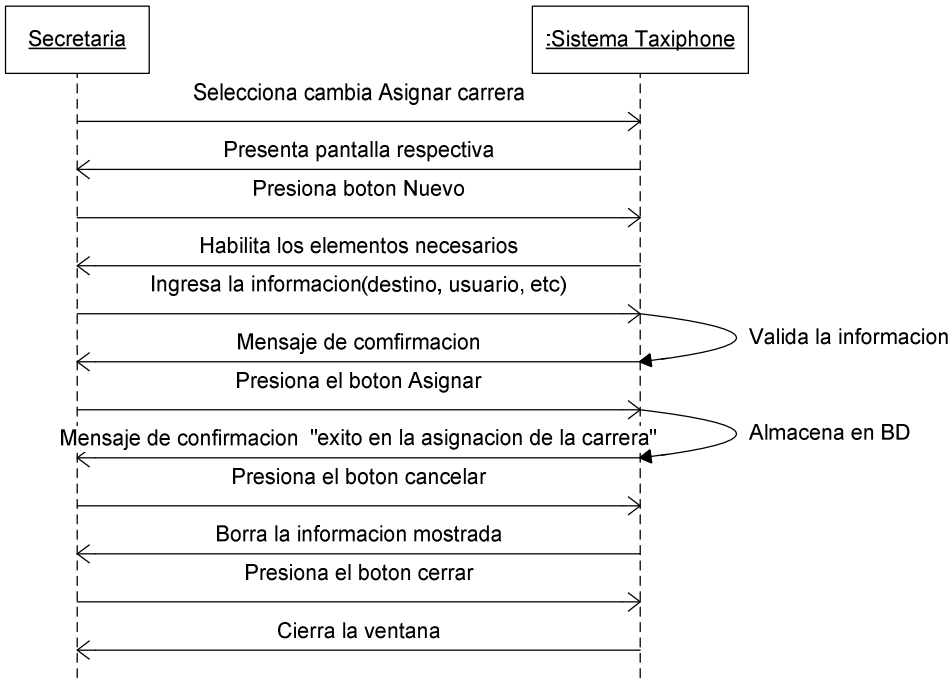


Figura Nº IV. 110. Diagrama de secuencia refinado: Asignar carrera

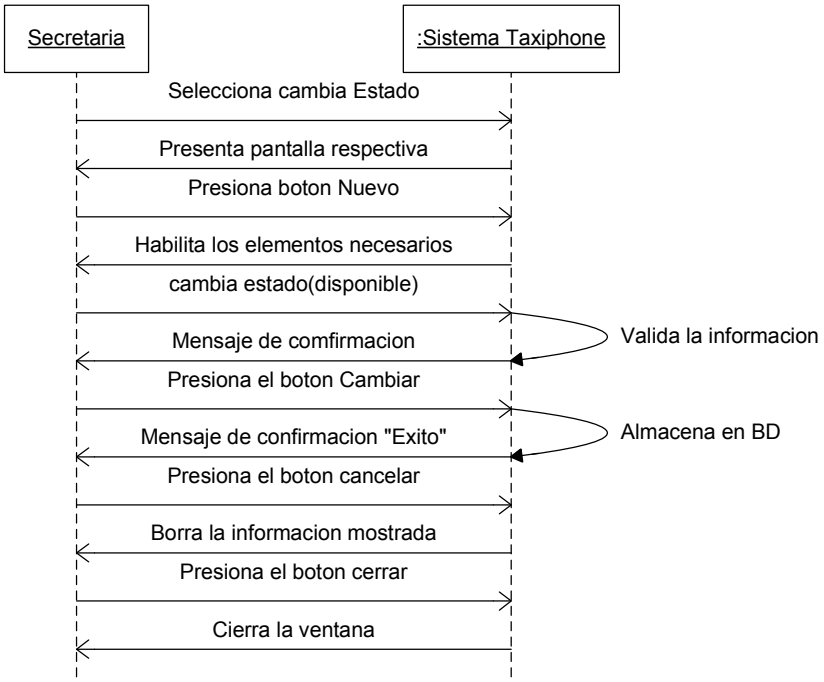


Figura Nº IV. 111. Diagrama de secuencia: Cambiar estado

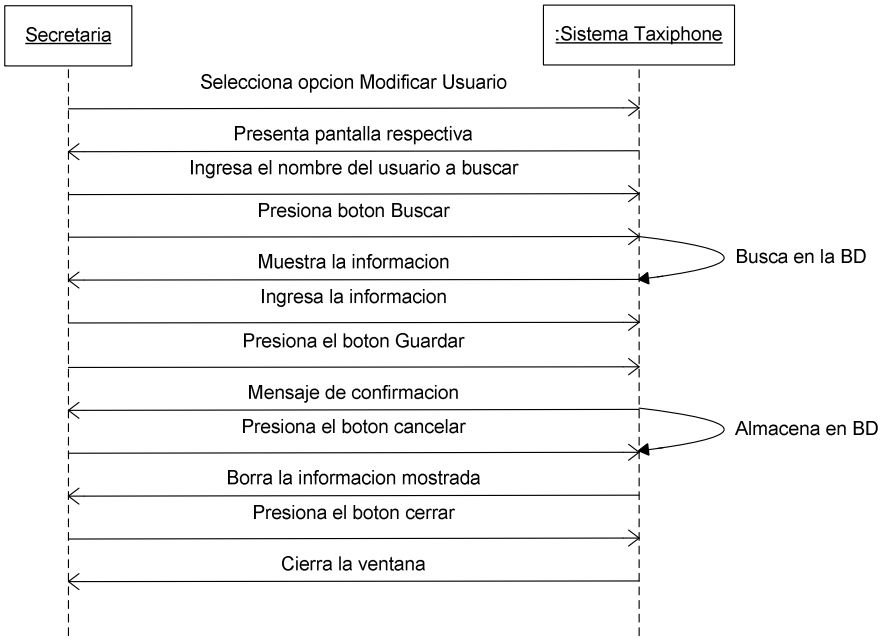


Figura Nº IV. 112. Diagrama de secuencia: Modificar usuario

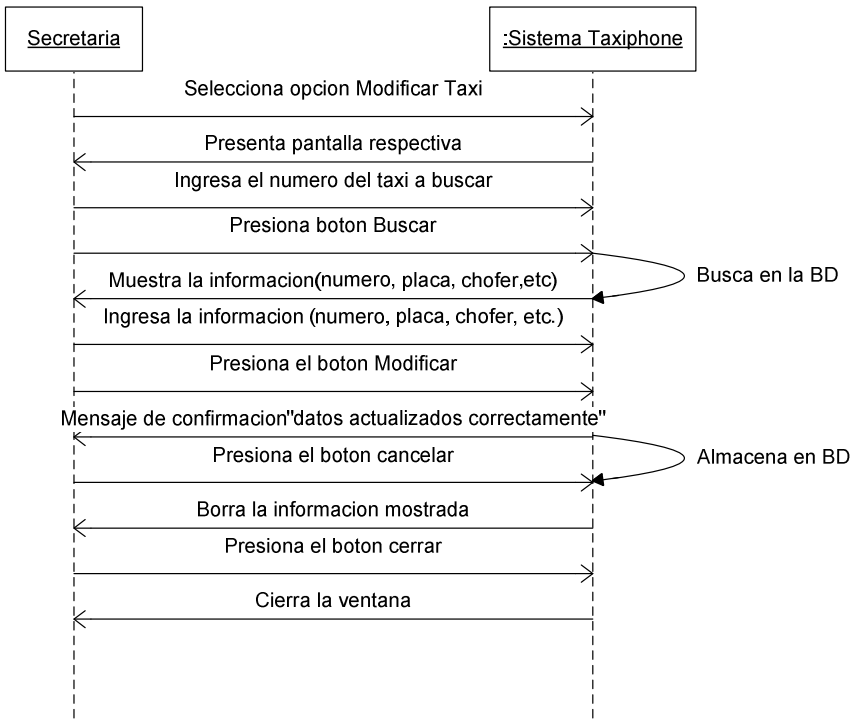


Figura Nº IV. 113. Diagrama de secuencia refinado: Modificar Taxi

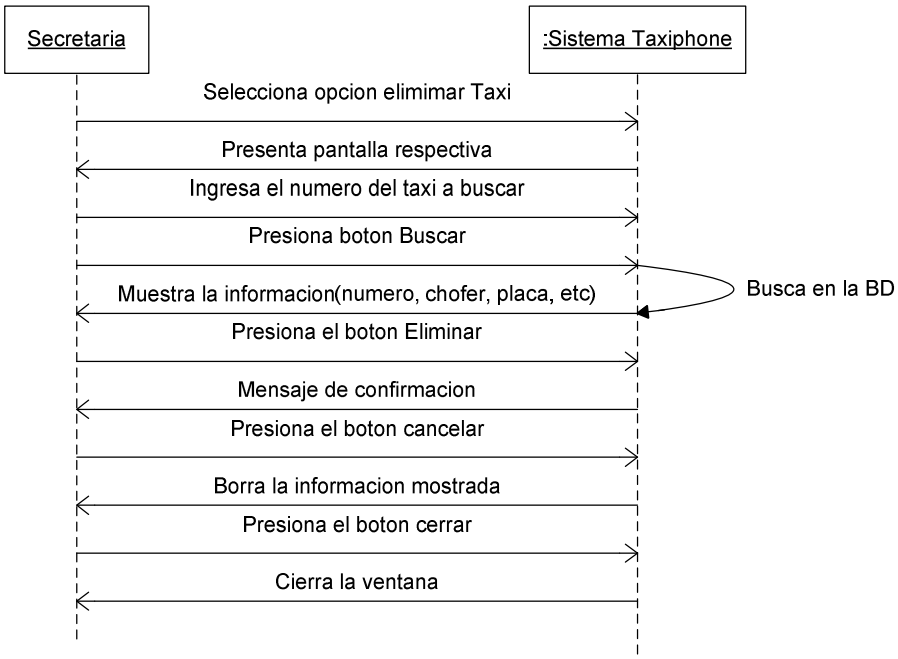


Figura Nº IV. 114. Diagrama de secuencia refinado: Eliminar taxi

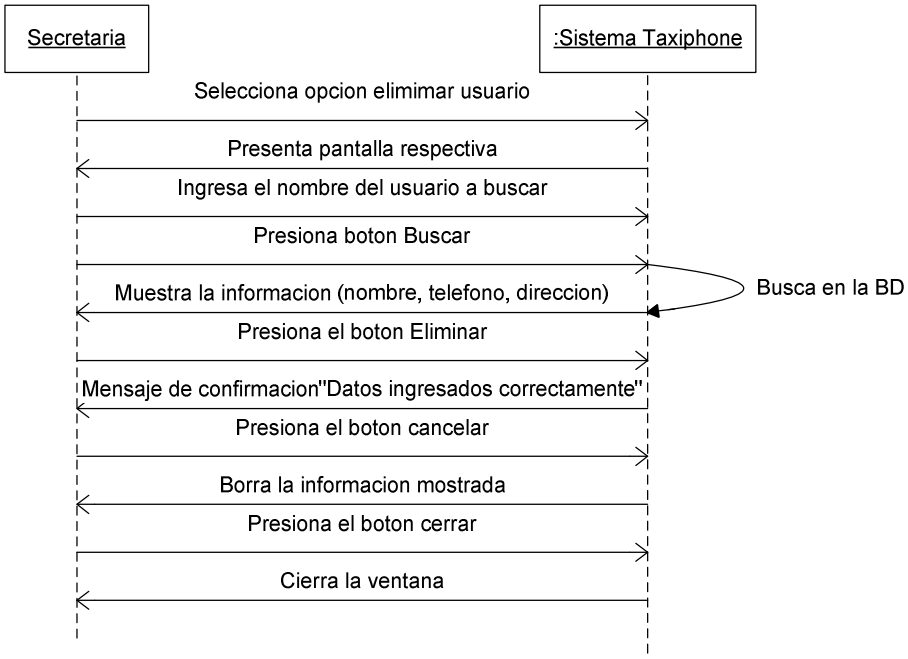


Figura Nº IV. 115. Diagrama de secuencia refinado: Eliminar usuario

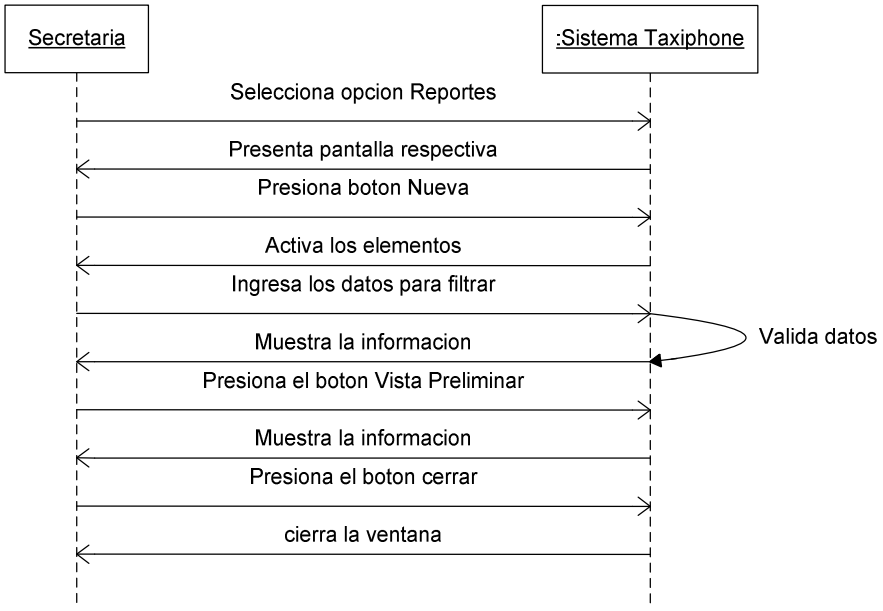


Figura Nº IV. 116. Diagrama de secuencia refinado: Reportes

4.2.1.3.2. Diagramas de clases refinados

En la figura Nº IV.117, se presenta el diagrama de clases refinado en el que se especifica, los atributos, tipos de datos, propiedades, clases, relaciones entre ellas, en mismo que servirá como punto de partida para crear el diagrama entidad relación de la base de datos tomando en consideración cada uno de estos aspectos

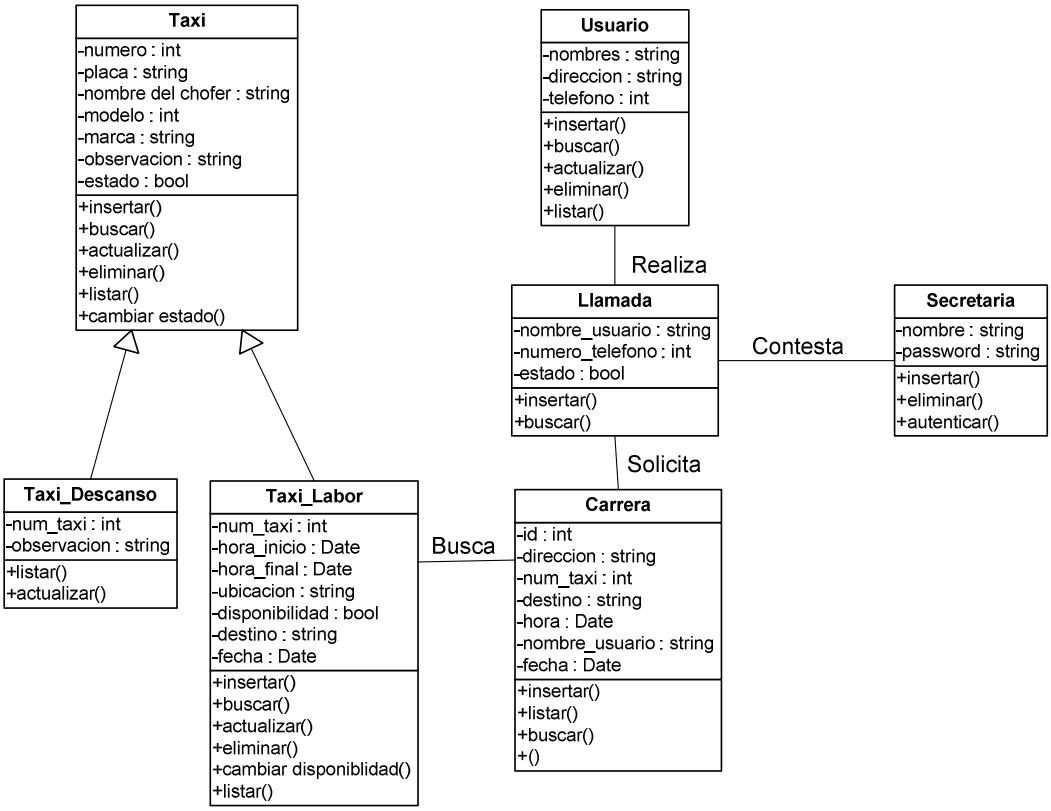


Figura Nº IV. 117. Diagrama de clases refinada

4.2.1.3.3. Diagramas de actividades

En la figura Nº IV.118, se ilustra en diagrama de actividades que cada una de las entidades realiza con relación al sistema taxiphone, en el que se trata de englobar el proceso de funcionamiento del sistema ante cada petición del usuario

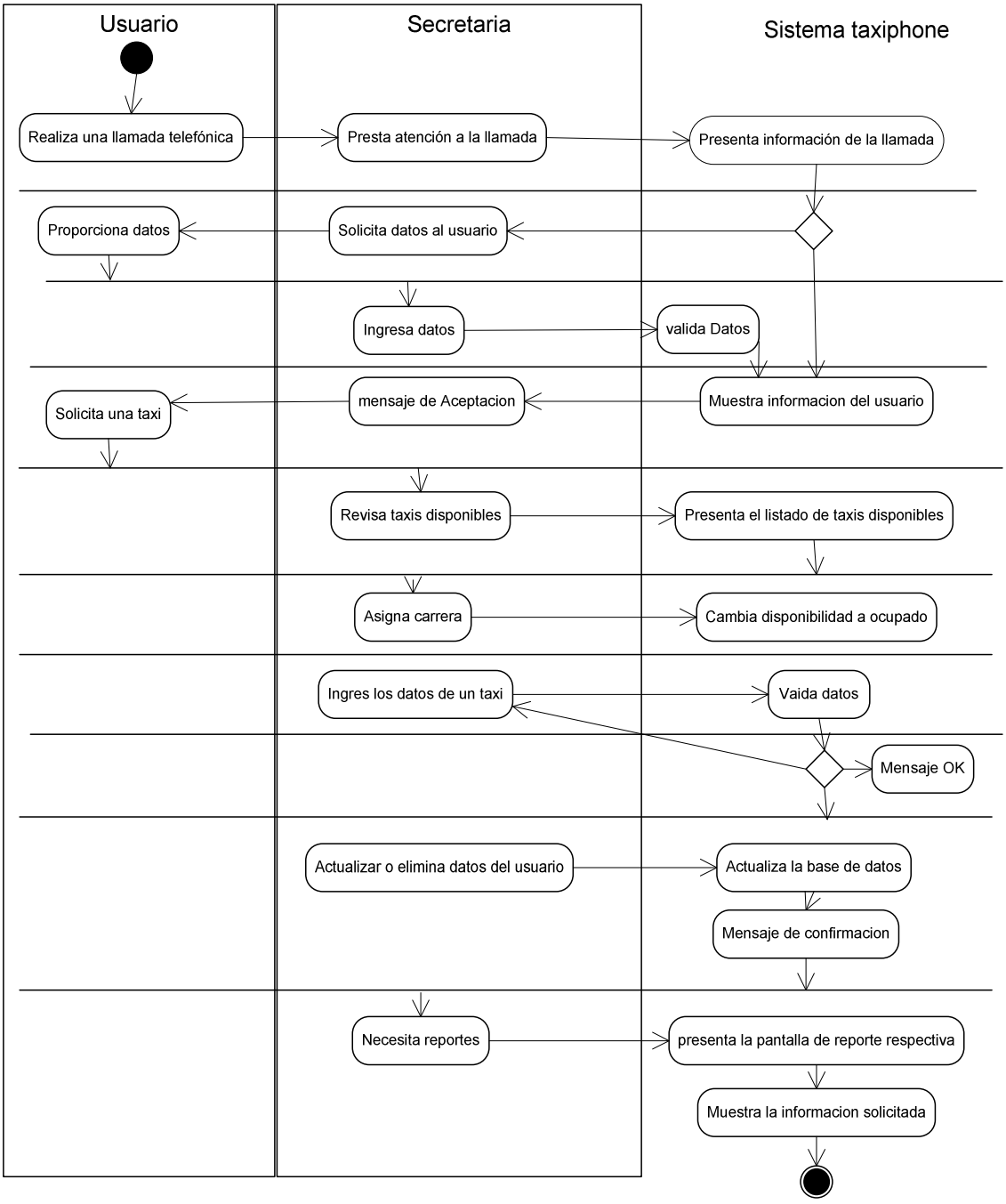


Figura Nº IV. 118. Diagrama de actividades

4.2.1.3.4. Diagramas de componentes

En la figura NºIV.119, se presenta el diagrama de componentes respectivo del sistema taxiphone

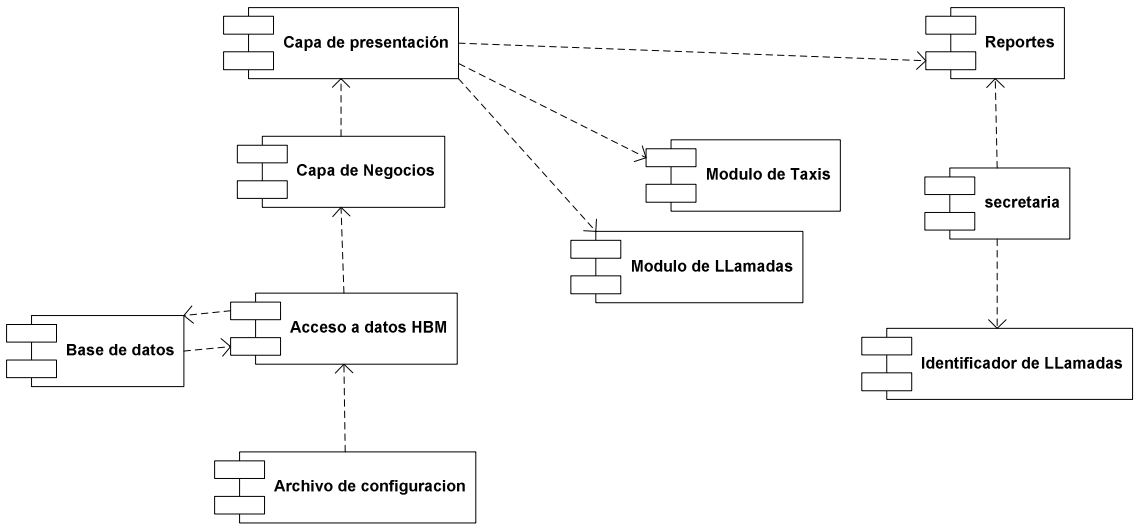


Figura Nº IV. 119. Diagrama de Componentes

4.2.1.3.5. Diagramas de Implementación

El diagrama del implementación se presenta en la figura Nº IV.120, en donde interviene la secretaria como usuario del sistema Taxiphone, la base de datos a la que se encuentra conectada directamente el sistema, los taxis, y los usuarios de la cooperativa, que solicitan los taxis, a través de la llamada telefónica.

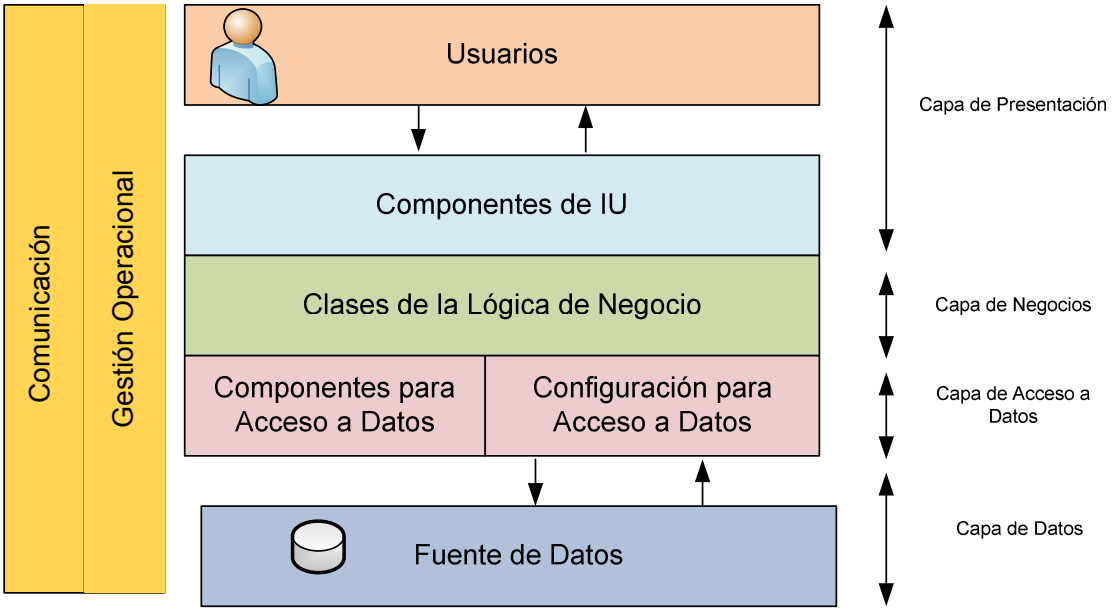


Figura Nº IV. 120. Diagrama de Implementación

4.2.1.3.6. Modelo físico de la Base de Datos

El diagrama entidad relación de la base de datos, obtenido en el SQL Server 2005, se plasma como modelo físico de la base de datos, el mismo que se lo presenta en la figura Nº IV.121, en el que se especifica las tablas, campos, llaves primarias, tipos de datos y relaciones entre ellas, la misma que servirán para tener referencia, el momento de la configuración en los archivos de mapeo.

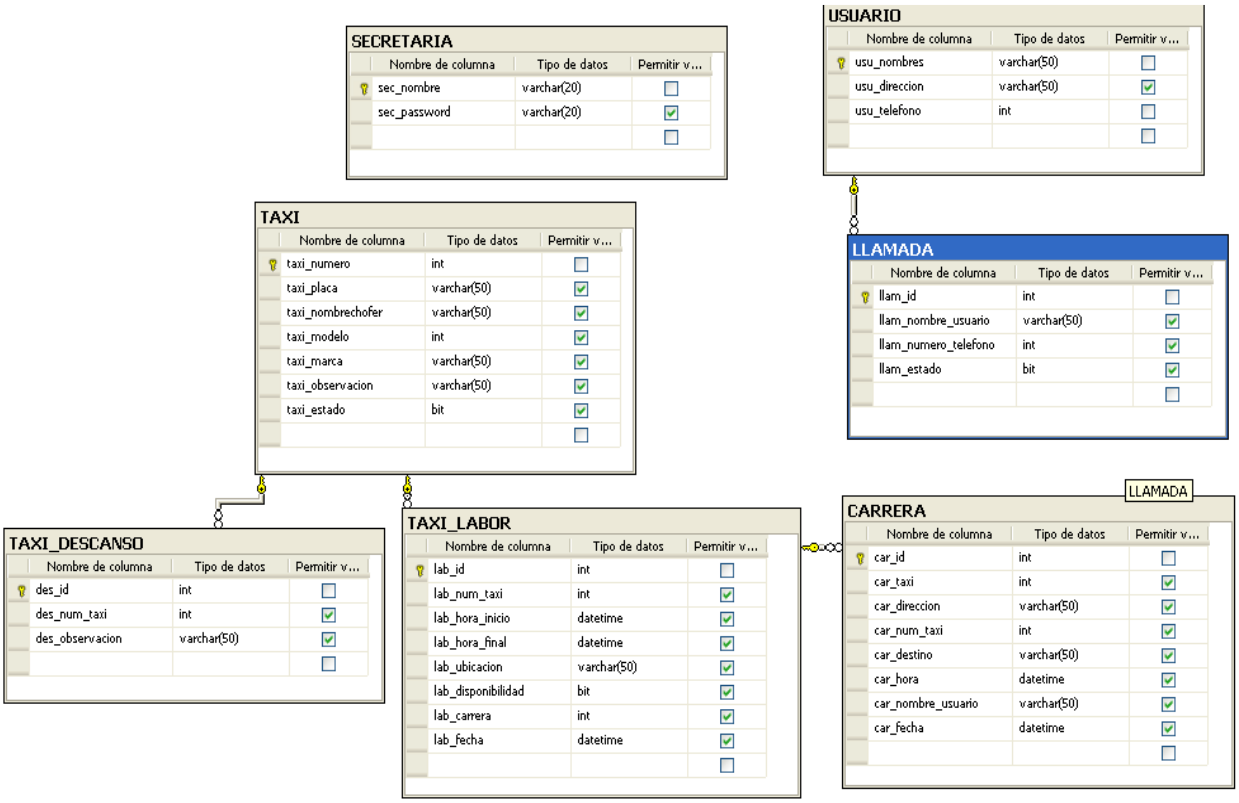


Figura Nº IV. 121. Modelo físico de la base de datos

4.3. DESARROLLO

Esta fase estará enfocada a los programadores que les servirá como guía para conocer el proceso del desarrollo de programación, en el que se centra su estudio, al código en cada una de sus capas de diseño de datos independientemente del lenguaje de programación inclusive se presentará el script de la base de datos, contemplando de esta manera la estructura abierta para dar a conocer el procedimiento de desarrollo, ésta fase irá de la mano con la fase de planeación y de estabilización para obtener versiones del producto que permitan brindar al programador una idea clara del sistema

4.3.1. Nomenclatura y Estándares para el Desarrollo

A continuación se establece los estándares de codificación del lenguaje, el mismo que será aplicado a todos los productos de la aplicación, tanto para la capa de acceso a datos, cada de negocios, capa de presentación, etc., como se puede observar en la tabla Nº IV.20

Tabla Nº IV. 20. Nomenclatura y estándares para el desarrollo

TIPO DE ARCHIVO	EXTENSIÓN	NOMENCLATURA
Archivo de mapeo	.hbm	NombreArchivo.hbm
Archivo de clase	.cs	NombreClase.cs
Archivo de diagramas de clase	.cd	NombreDiagrama.cd
Archivo de ayuda	.chm	NombreAyuda.chm
Archivo de configuración	.cfg.xml	NombreArchivo.cfg.xml

Archivo HBM

Los hbm son archivos de mapeo de hibernate,son de tipo xml y representan a una clase y tabla determinada. Haremos un ejemplo para ilustrar esta explicación.

Archivo CS

Son los archivos que contienen código del lenguaje C#, pueden tener esta extensión los archivos tipo clase, ó los formularios winddows

Archivo CHM

Archivos que poseen información de ayuda de las aplicaciones que se están usando.

Archivo CFG.XML

Archivos de configuración de tipo xml, que poseen código para configurar la conexión de NHibernate

Archivo CD

Extensión de los archivos que se les proporciona a los diagramas de clases del lenguaje C#

4.3.2. Capa de Presentación

En la capa de presentación se brindará el prototipo de las interfaces aprobadas para el sistema Taxiphone, en el que se engloba, los colores, navegabilidad, manejo, presentación de mensajes y de información; las mismas que se encuentran acordes a la función por la que es desarrollado el sistema.

4.3.2.1. Diseño de interfaces de usuario

A continuación se presentan las interfaces que se tomarán en consideración en el sistema Taxiphone, que involucran, pantallas de bienvenida, autenticación, inserción, actualización, eliminación, listados y reportes, las mismas que se ilustran a partir de la figura Nº IV.122 hasta la figura Nº IV.135

Figura Nº IV. 122. Pantalla de Bienvenida

Figura Nº IV. 123. Pantalla de Autenticación

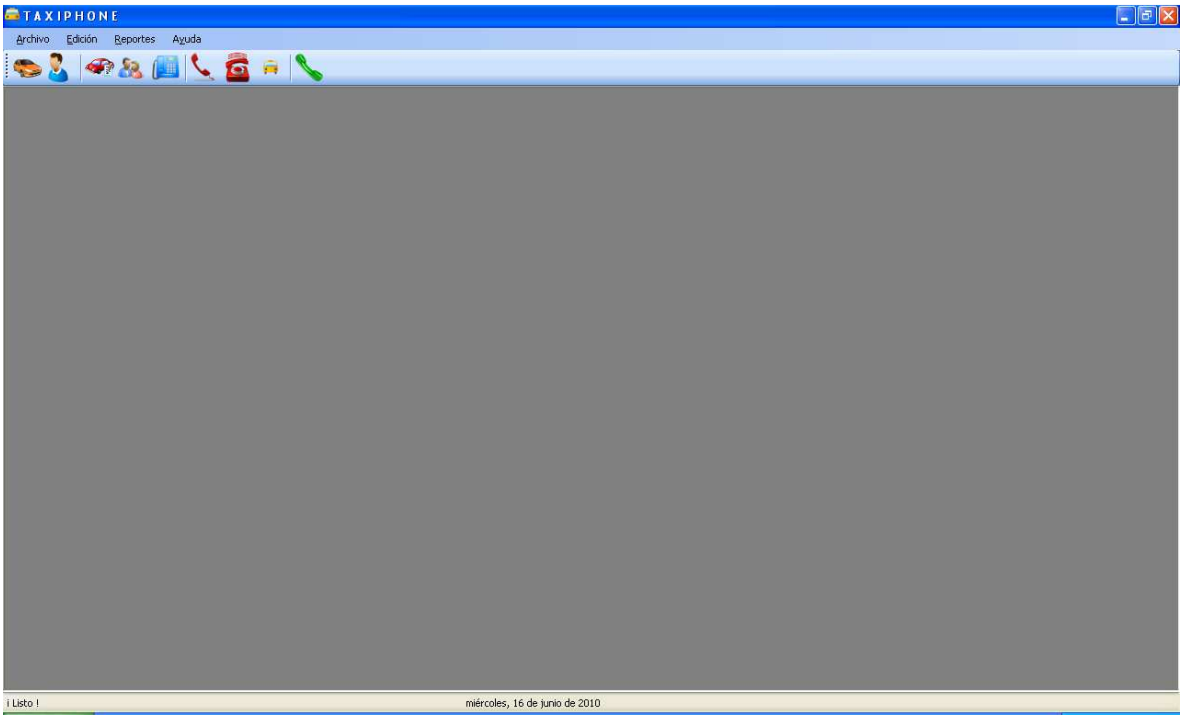


Figura Nº IV. 124. Pantalla principal

Figura Nº IV. 125. Pantalla para insertar Usuario

Figura Nº IV. 126. Pantalla para insertar taxi

Figura Nº IV. 127. Pantalla para cambiar el estado del taxi

Figura Nº IV. 128. Pantalla para asignar carrera

Figura Nº IV. 129. Pantalla para cambiar la disponibilidad del taxi

Figura Nº IV. 130. Pantalla para atender una llamada entrante

Figura Nº IV. 131. Pantalla para modificar al usuario

Figura Nº IV. 132. Pantalla para buscar un taxi

Figura Nº IV. 133. Pantalla para eliminar usuario

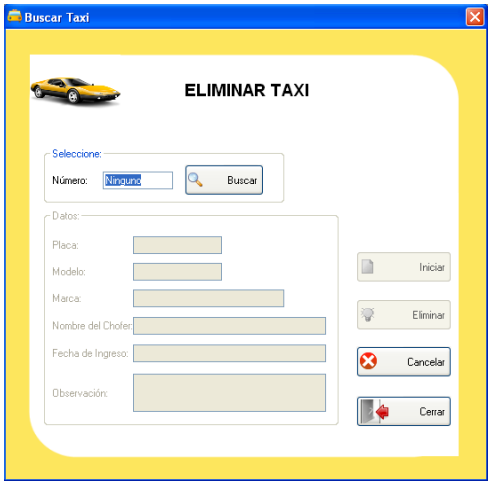


Figura Nº IV. 134. Pantalla para eliminar un taxi

Figura Nº IV. 135. Pantalla para listar todos los usuarios

4.3.3. Capa de Datos

En la fase de desarrollo de la capa de datos, se logrará detallar la implementación que se realiza en cada capa de desarrollo, indicando así el proceso, comunicación entre cada capa, el objetivo o fin de cada una de ellas, y medidas de seguridad que se toman para conseguir un sistema de calidad

4.3.3.1. Implementación de la base de datos

La implementación de la base de datos se lo realizó en el motor de BD SQL Server 2005, la base de datos cuenta con siete tablas relacionadas entre ellas, cada una tiene su llave primaria que las identifica, la base de datos no requiere de procedimientos almacenados ya que éste tipo de consultas se lo realizará a través de código en capa de negocios.

4.3.3.2. Implementación de Acceso a Datos

Como se mencionaba anteriormente la implementación de acceso a datos es la parte esencial de NHibernate, por lo tanto es aquí en donde se logra configurar el mapeo de cada objeto, los archivos de acceso a datos se caracterizan por contener el código xml, y trabajan con la extensión .hbm, que permiten identificarlos claramente, estos archivos contienen la estructura de cada objeto/clase, lo que permite vincular entre la capa de negocios con la base de datos.

4.3.4. Capa de Negocios

En la capa de negocios se detalla, la composición de los elementos que conforman el sistema, entre ellos, los componentes que se encuentran en cada capa, y la relación que existen entre ellos.

4.3.4.1. Implementación de componentes

En la figura Nº IV.136, se muestra los componentes que conforman el sistema, incluyendo la relación que existe entre ellos.

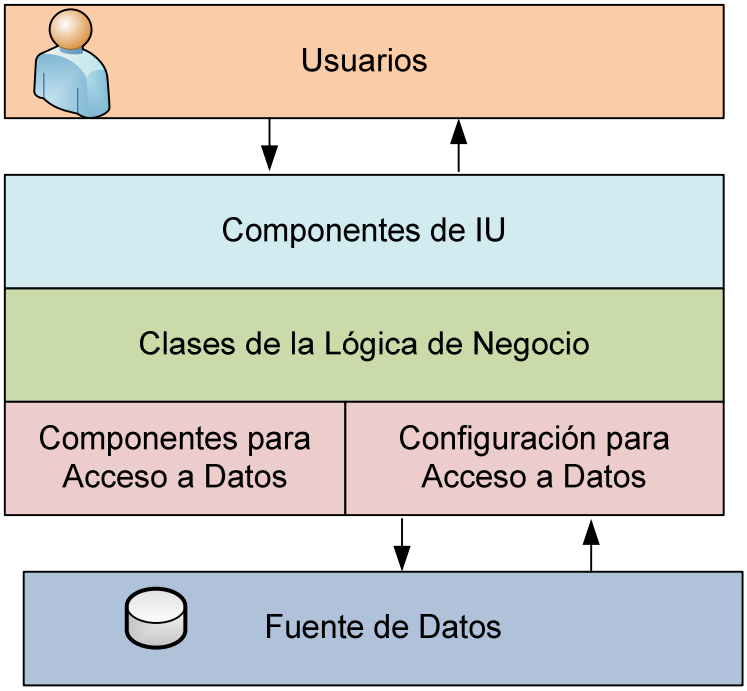


Figura Nº IV. 136. Componentes del sistema

4.3.5. Especificaciones de Seguridad

Las estrategias de seguridad se plantea de acuerdo a los permisos de autenticación, autorizaciones o auditorias que se ejecutan en cuanto al funcionamiento del sistema Taxiphone.

4.3.5.1. Diseño y estrategias de autorización, autenticación y auditoria

La seguridad planteada en base a la autenticación de usuarios, está restringida al uso, ya que evita que cualquier persona tenga acceso al sistema, sino se encuentra registrado como usuario del mismo.

Las autorizaciones se establecen, en cuanto a permisos de funcionamiento, o licenciamiento del sistema, ya que el sistema Taxiphone bridará mejoras a la cooperativa Taxiphone en base a sus necesidades.

La auditoría se lo realizará en cuanto al funcionamiento, para la verificación y cumplimiento de todas las necesidades de la cooperativa sean cubiertas, de manera que garantice la calidad y disponibilidad del sistema hacia la cooperativa

4.4. ESTABILIZACIÓN

En la fase de estabilización se realizará las pruebas, correcciones y se proporcionará los códigos fuentes del sistema que servirá como soporte de ayuda para los programadores cuando se requiera ejecutar cambios en el sistema.

4.4.1. Revisión General del Sistema

Para realizar la revisión general del sistema, se procede a presentar el código fuente que posee el sistema, tanto de la aplicación como de la base de datos

4.4.1.1. Código fuente y ejecutable

Código para la Autenticación

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NHibernate;
using NHibernate.Dialect;
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;

namespace radiotaxi
{
   public partial class FrmAutenticar : Form
   {
      public FrmAutenticar()
      {
         InitializeComponent();
      }

      /// <summary>
/// Inserta a un usuario dado los parametros de la clase usuario
/// </summary>
/// <param name="usuario"></param>
      public static void autenticacion(Secretaria secretaria)
      {
ISession getNewS = clsPrincipal.GetNewSession();
         using (NHibernate.ISession sessionTwo = getNewS)
         {
            using (NHibernate.ITransaction tx = sessionTwo.BeginTransaction())
            {
               try
               {
                  sessionTwo.Lock(secretaria, LockMode.None);
                 tx.Commit();
               }
               catch
               { tx.Rollback(); }
               sessionTwo.Close();
               clsPrincipal.CloseSessionFactory();
            }
         }

      }

      private void FrmAutenticar_Load(object sender, EventArgs e)
      {
         ControladorInterfaz(1);
      }

      private void btnAceptar_Click(object sender, EventArgs e)
      {
         Secretaria objsecretaria=new Secretaria();
         objsecretaria.SecNombre=this.txtNombre.Text;
         objsecretaria.SecPassword=txtNombre.Text;
         if (objsecretaria.SecNombre == txtNombre.Text && objsecretaria.SecPassword==
txtContrasenia.Text)
         {
            autenticacion(objsecretaria);
            this.Hide();
          //creamos una instancia de la forma
         FrmPrincipal objfrmPrincipal= new FrmPrincipal();
```

```
            objfrmPrincipal.Show();//mostramos la forma instanciada
            }
            else MessageBox.Show("Datos incorrectos, acceso Denegado");
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            ControladorInterfaz(3);
        }

        private void btnCerrar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void txtNombre_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (Char.IsDigit(e.KeyChar) == true)//detecta si se ingresa números
            {
                MessageBox.Show("No puede ingresar números", "Mensaje", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                e.Handled = true;//no pase ese caracter al textbox q se acaba de ingresar
            }
        }

        private void txtNombre_Validating(object sender, CancelEventArgs e)
        {
            if ((this.txtNombre.Text == "Ninguno") || (this.txtNombre.Text == ""))//si no se ha ingresado datos
            { //MessageBox.Show("Ingrese el nombre");
                e.Cancel = true;//cancelece Para q pase al siguiente control

                this.erpError.SetError(this.txtNombre, "Ingrese el Nombre");//muestra un mensaje de error
            }
            else
                this.erpError.SetError(this.txtNombre, "");
        }

        private void txtContrasenia_Validating(object sender, CancelEventArgs e)
        {
            if ((this.txtContrasenia.Text == "Ninguno") || (this.txtContrasenia.Text == ""))//si no se ha ingresado
datos
            { //MessageBox.Show("Ingrese el nombre");
                e.Cancel = true;//cancelece Para q pase al siguiente control
                this.erpError.SetError(this.txtContrasenia, "Ingrese la Contraseña");//muestra un mensaje de
error
            }
            else
                this.erpError.SetError(this.txtContrasenia, "");
        }
    }
}
```

Código de la Clase Principal

```
using System;
using System.Collections.Generic;
using System.Text;
using NHibernate;
using NHibernate.Dialect;
```

```
using NHibernate.Mapping;
using NHibernate.SqlCommand;
using NHibernate.Cfg;
using NHibernate.Collection;
using NHibernate.Tool.hbm2ddl;
using NHibernate.Expression;


namespace radiotaxi
{
  public class clsPrincipal
   {
        private static ISessionFactory SessionF;
      /// <summary>
      /// configuracion de la cadena de conexion
      /// </summary>
          private static void Init()
          {
              Configuration config = new Configuration();
              config.Configure("E:\\Documents and Settings\\Ivito\\Escritorio\\APLICACION para
modif\\radiotaxi\\radiotaxi\\hibernate.cfg.xml");
              SessionF = config.BuildSessionFactory();
          }


          public static ISessionFactory GetSessionFactory()
          {
              if (SessionF == null)
                  Init();
              return SessionF;
          }

/// <summary>
/// abre la sesion
/// </summary>
/// <returns></returns>
          public static ISession GetNewSession()
          {
              return GetSessionFactory().OpenSession();
          }

          /// <summary>
          /// cierra la sesion
          /// </summary>
          public static void CloseSessionFactory()
          {
              if (GetSessionFactory() != null)
                  GetSessionFactory().Close();
          }
      }
  }
```

Código de la Clase Taxi

```
using System.Collections.Generic;
using System.Text;
using System;

namespace radiotaxi
{
```

```
    public class Taxi
    {
        //declaracion de variables
        private int taxiNumero;
        private string taxiPlaca;
        private string taxiChofer;
        private int taxiModelo;
        private string taxiMarca;
        private string taxiObservacion;
        private Boolean taxiEstado;
        private IList<Taxi_Descanso> descanso;
        private IList<Taxi_Labor> labor;

        /// <summary>
        /// constructor de la clase TAXI
        /// </summary>
        /// <param name="taxiNumero"></param>
        /// <param name="taxiPlaca"></param>
        /// <param name="taxiChofer"></param>
        /// <param name="taxiModelo"></param>
        /// <param name="taxiMarca"></param>
        /// <param name="taxiObservacion"></param>
        /// <param name="taxiEstado"></param>
        public Taxi(int taxiNumero,string taxiPlaca,string taxiChofer,int taxiModelo,string taxiMarca,string
taxiObservacion,Boolean taxiEstado):this()
        {
            this.taxiNumero = taxiNumero;
            this.taxiPlaca = taxiPlaca;
            this.taxiChofer = taxiChofer;
            this.taxiModelo = taxiModelo;
            this.taxiMarca = taxiMarca;
            this.taxiObservacion = taxiObservacion;
            this.taxiEstado = taxiEstado;
        }
        public Taxi()
        {
            descanso = new List<Taxi_Descanso>();
            labor = new List<Taxi_Labor>();
        }

        public virtual int TaxiNumero
        {
            get { return taxiNumero; }
            set { taxiNumero = value; }
        }

        public virtual string TaxiPlaca
        {
            get { return taxiPlaca; }
            set { taxiPlaca = value; }
        }

        public virtual string TaxiChofer
        {
            get { return taxiChofer; }
            set { taxiChofer = value; }
        }

        public virtual int TaxiModelo
        {
            get { return taxiModelo; }
            set { taxiModelo = value; }
        }
```

```
        public virtual string TaxiMarca
        {
            get { return taxiMarca; }
            set { taxiMarca = value; }
        }

        public virtual string TaxiObservacion
        {
            get { return taxiObservacion; }
            set { taxiObservacion = value; }
        }

        public virtual bool TaxiEstado
        {
            get { return taxiEstado; }
            set { taxiEstado = value; }
        }

        public virtual IList<Taxi_Labor> Labor
        {
            get { return labor; }
            set { labor = value; }
        }

        public virtual IList<Taxi_Descanso> Descanso
        {
            get { return descanso; }
            set { descanso = value; }
        }

    }
}
```

Código del archivo de mapeo Taxi

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.0" namespace="radiotaxi" assembly="radiotaxi">
  <class name="Taxi" table="TAXI">
    <id name="TaxiNumero">              <column name="taxi_numero" sql-type="int" not-null="true"
length="4" />   <generator class="uuid.hex" />    </id>
    <property name="TaxiPlaca">          <column name="taxi_placa" />        </property>
    <property name="TaxiChofer">         <column name="taxi_nombrechofer" />  </property>
    <property name="TaxiModelo">          <column name="taxi_modelo" />       </property>
    <property name="TaxiMarca">          <column name="taxi_marca" />        </property>
    <property name="TaxiObservacion">     <column name="taxi_observacion" />   </property>
    <property name="TaxiEstado">         <column name="taxi_estado" />       </property>
    <set name="Descanso" lazy="true" cascade="delete">
      <key column="des_num_taxi" />
      <one-to-many class="Taxi_Descanso" />
    </set>
    <set name="Labor" lazy="true" cascade="delete">
      <key column="lab_num_taxi" />
      <one-to-many class="Taxi_Labor" />
    </set>
  </class>
</hibernate-mapping>
```

4.4.1.2. Scripts de base de datos

El script de la base de datos se presenta en el siguiente código fuente, extraido desde SQL Server 2005.

```
CREATE DATABASE [radiotaxi] ON  PRIMARY
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [CARRERA](
	[car_id] [int] NOT NULL,
	[car_direccion] [varchar](50) NULL,
	[car_num_taxi] [int] NULL,
	[car_destino] [varchar](50) NULL,
	[car_hora] [datetime] NULL,
	[car_nombre_usuario] [varchar](50) NULL,
	[car_fecha] [datetime] NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [LLAMADA](
	[llam_nombre_usuario] [varchar](50) NOT NULL,
	[llam_numero_telefono] [int] NULL,
	[llam_estado] [bit] NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [SECRETARIA](
	[sec_nombre] [varchar](20) NOT NULL,
	[sec_password] [varchar](20) NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [TAXI](
	[taxi_numero] [int] NOT NULL,
	[taxi_placa] [varchar](50) NULL,
	[taxi_nombrechofer] [varchar](50) NULL,
	[taxi_modelo] [int] NULL,
	[taxi_marca] [varchar](50) NULL,
	[taxi_observacion] [varchar](50) NULL,
	[taxi_estado] [bit] NULL
) ON [PRIMARY]

GO
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [TAXI_DESCANSO](
	[des_num_taxi] [int] NOT NULL,
	[des_observacion] [varchar](50) NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [TAXI_LABOR](
	[lab_num_taxi] [int] NOT NULL,
	[lab_hora_inicio] [datetime] NULL,
	[lab_hora_final] [datetime] NULL,
	[lab_ubicacion] [varchar](50) NULL,
	[lab_disponibilidad] [bit] NULL,
	[lab_destino] [varchar](50) NULL,
	[lab_fecha] [datetime] NULL
) ON [PRIMARY]

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [USUARIO](
	[usu_nombres] [varchar](50) NOT NULL,
	[usu_direccion] [varchar](50) NULL,
	[usu_telefono] [int] NOT NULL
) ON [PRIMARY]
```

4.4.1.3. Documentación de la instalación

El sistema taxiphone, cuenta con la ayuda necesaria para conseguir su correcto funcionamiento, en lo que se incluye la ayuda en línea, manual técnico y de usuario, capacitación y soporte técnico al personal que trabajará directamente con el sistema taxiphone

4.4.1.4. Material de capacitación

El material de capacitación está conformado por los manuales de usuario y técnico que servirán como guía para los usuarios y técnicos del sistema, evitando de ésta manera inconvenientes para el correcto funcionamiento del mismo.

La capacitación está destinada exclusivamente a todo el personal que pertenece a la cooperativa Taxiphone, identificando el proceso de funcionamiento del mismo, e inclusive dar a conocer el beneficio que otorgará a dicha cooperativa.

4.4.1.5. Historial de versiones

Las versiones del sistema está determinado en la tabla Nº IV.21, indicando el tipo de versión la fecha que se lo ha realizado y los motivos por los cuales a variado la versión.

Tabla Nº IV.21. Tabla de versiones de sistema

VERSIÓN	FECHA	CAUSA
1.0.0	Abril del 2009	Bosquejo de las interfaces a realizarse
1.0.1	Junio del 2009	Modificación en las interfaces y control de las mismas
1.1.0	Agosto 2009	Conexión con la base de datos, inserción de la información
1.1.1	Octubre del 2009	Control en la inserción y abstracción de datos
1.2.0.	Diciembre del 2009	Generación de reportes con Crystal Report
1.3.0.	Enero del 2010	Entrega final del sistema

4.4.2. Pruebas

Las pruebas ejecutadas en el sistema se lo realizaron en cuanto a la conexión con la base de datos, inserción y abstracción de datos, en la compilación y manejo de la información.

Estas pruebas de desarrollo están establecidas con el fin de proporcionar estabilidad, fiabilidad y soporte en el manejo de la información.

Se realizaron pruebas de diseño de interfaz, en el que se propone un acuerdo en los usuarios y el manejo del tipo de información, cabe mencionar que se debió realizar controles, ayudas y cuadros de diálogos que brinden seguridad en el manejo del sistema

4.5. INSTALACIÓN

La instalación del sistema se realizará en las computadoras, que posee la cooperativa, que será manipulado directamente por las secretarias.

La instalación se realizará del motor de base de datos SQL Server 2005, y del ejecutable del sistema Taxiphone, dejándolo en correcto funcionamiento.

Para comprobar la solidez y estabilidad del sistema, se trabajo en el escenario real, para determinar posibles falencias del sistema, que deberán ser controlados a tiempo

CAPÍTULO V

ANÁLISIS DE RESULTADOS

En el presente capítulo se realiza el estudio y análisis de resultados obtenidos a lo largo del estudio del framework NHibernate y del desarrollo de la aplicación Windows para la detección de llamadas en la cooperativa de taxis Taxiphone; para lograr la comprobación de la hipótesis, se aplicó las técnicas de las encuestas y entrevistas a usuarios y taxistas respectivamente, además de ello se aplicó la observación de campo, para lo cual se requiere plantear y ejecutar la operacionalización conceptual y metodológica de variables, identificándolas desde la hipótesis investigativa; para conseguir la comprobación final de la misma, se procede a realizar una escala de valores entre las variables, las cuales servirán para realizar la graficación comparativa y estadística de cada variable independiente que servirán de apoyo para sustentar la demostración de la hipótesis planteada previo la instalación del sistema Taxiphone.

5.1.Descripción de la Hipótesis

Hipótesis investigativa: "La aplicación del Framework de persistencia NHIBERNATE en el sistema de detección de llamadas mejorará en tiempo y seguridad la atención a los usuarios de la cooperativa TAXIPHONE"

Operacionalización Conceptual de Variables:

En la Tabla V.22, se muestra la operacionalización conceptual de variables, en la que se detalla las variables dependientes e independientes

Tabla V.22. Operacionalización Conceptual de Variables

VARIABLE	TIPO DE VARIABLE
La aplicación del framework de persistencia NHibernate en el sistema de detección de llamadas	V. Independiente
Mejorar en tiempo y seguridad la atención a los usuarios de la cooperativa TAXIPHONE	V. Dependiente

Operacionalización Metodológica de Variables:

En la Tabla V.23 se muestra la Operacionalización Metodológica de Variables, con la categoría, indicadores, técnicas y fuentes de verificación en base a las variables planteadas.

Tabla V.23. Operacionalización Metodológica de Variables

VARIABLE	CATEGORÍA	INDICADORES	TÉCNICA	FUENTE DE VERIFICACIÓN
La aplicación del framework NHibernate en el sistema de detección de llamadas		El uso de la aplicación NHibernate en el sistema de detección de llamadas	Observación	Software Taxiphone Documentación técnica del sistema
Mejorar en tiempo y seguridad la atención a los usuarios de la cooperativa TAXIPHONE	Tiempo	Tiempo de espera hasta llegar el taxi a su destino	Encuestas Observación de campo	Sede de la cooperativa
	Seguridad	Seguridad para los usuarios	Encuestas	Lugar en donde se encuentra el usuario

5.2. Comprobación dé Hipótesis

Para lograr la comprobación de la hipótesis, se aplicó las técnicas de investigación entre ellas las encuestas a taxistas y usuarios, entrevistas a la secretaria y la observación de campo

Para obtener la muestra de la población a la que se les aplicará las encuestas, se consideró el tamaño de la población.

La población de taxistas es de 35 por tanto se tomará como muestra toda la población, a diferencia de la población de usuarios que es desconocida o indeterminada, para aquello se aplicó la siguiente fórmula

$$n = \frac{z^2 * P * Q}{E^2}$$

En donde:

Z= es el coeficiente de confianza si es el 95% entonces z= 2.00

P= es la probabilidad con la que puede ocurrir, planteamos que el 80% ocurra

Q= es la probabilidad que no ocurra que es el 20%

E= Error admisible

Tomando en consideración que el error admisible es del 10% y el nivel de confianza del 95%

Con estos datos el resultado sería:

$$n = \frac{(2)^2 * 0.8 * 0.2}{(0.1)^2} = 64$$

La muestra de los usuarios es de 64, por tanto se aplicará todas las técnicas de investigación mencionadas con el sistema actual o manual y con el sistema

automatizado Taxiphone, para realizar la comparación respectiva de sistema a sistema se aplicó las técnicas contempladas en la Tabla Nº V.24

Tabla Nº V.24. Técnicas de investigación

TÉCNICA	PERSONAL	CANTIDAD
Encuesta	Usuario	64
Encuesta	Taxista	35
Entrevista	Secretaria	1
Observación	Usuario / Taxista	64

ENCUESTA A USUARIOS

A continuación se realiza el análisis de la encuesta aplicada a los usuarios de la cooperativa, la misma que se encuentra como Anexo Nº 1

Tabla Nº V.25. Zona de Riobamba en que habita.

ZONA DE RIOBAMBA	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Norte	12	19	19	30
Sur	18	28	15	23
Este	14	22	13	20
Oeste.	20	31	17	27
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

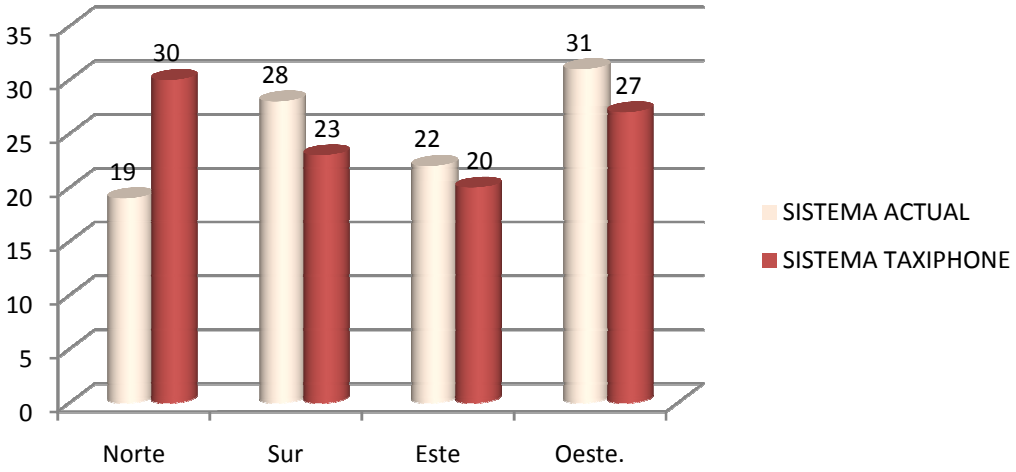


Figura Nº V. 137. Gráfico estadístico: Zona de Riobamba en que habita

Análisis

Según los datos presentados en la tabla Nº V.25, en donde se presenta una comparación entre los sistemas manual y Taxiphone, se puede determinar que los usuarios que habitan en la zona Oeste representados por el 31 % utilizaban el servicio del sistema manual de la cooperativa de taxi, a diferencia que en el sistema Taxiphone, los usuarios que la mayor parte utilizaban éste sistema son los que habitan en la zona Norte representado con el 30%, tal y como se presenta en la figura Nº V.137

TABLA Nº V.26. Frecuencia en que solicita el servicio de taxi de la Cooperativa.

FRECUENCIA	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Siempre	19	30	24	38
Frecuente	18	28	19	30
Rara vez	15	23	17	26
Nunca	12	19	4	6
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

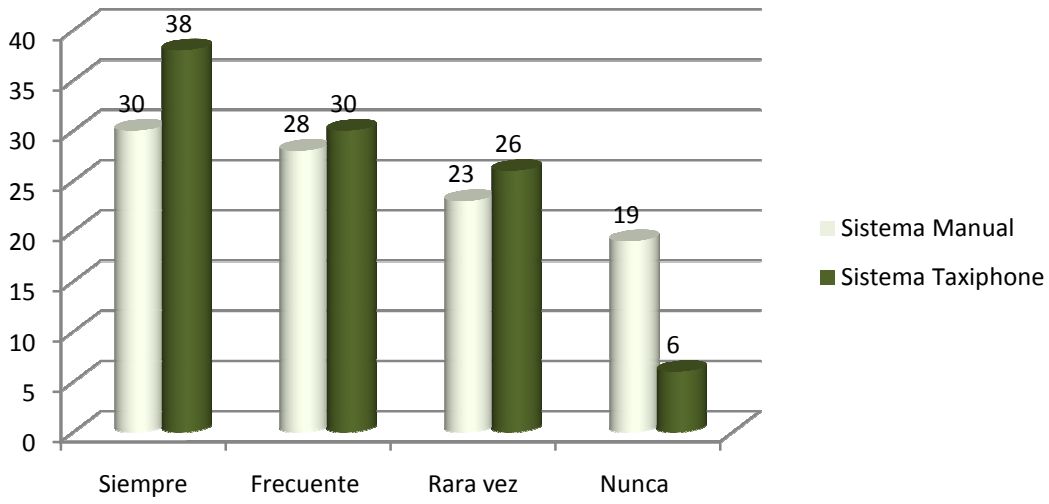


Figura Nº V. 138. Frecuencia en que solicita el servicio de taxi de la Cooperativa.

Análisis:

Al realizar el análisis comparativo entre los dos sistemas en base a los datos obtenidos presentados en la Tabla Nº V. 26 se pudo comprobar que, siempre solicitan el servicio de taxi en el sistema manual respondieron 19 usuarios equivalente al 30%, en cambio en el sistema manual respondieron 24 usuarios que siempre solicitan el servicio del taxi representado con el 38%, estos datos se encuentran ilustrados en la figura Nº V.138

TABLA Nº V.27. Cada que solicita el taxi siempre es atendido su carrera.

ATENCIÓN	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Si	39	61	53	83
No	25	31	11	17
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

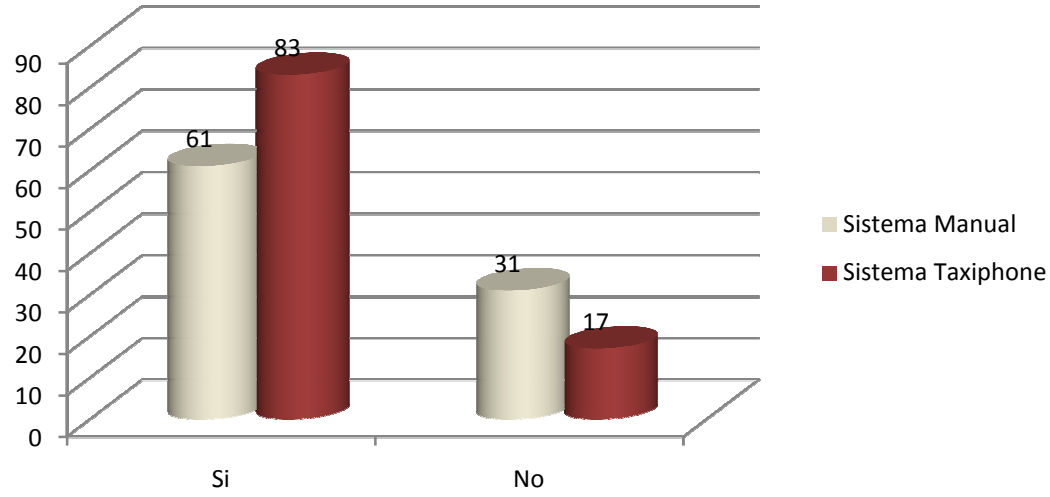


Figura Nº V. 139. Cada que solicita el taxi siempre es atendido su carrera

Análisis

Según la tabla Nº V.27, se presenta los datos comparativos entre el sistema manual y Taxiphone, de acuerdo a la atención que se le brinda a las carreras solicitadas, en donde 39 usuarios dijeron que si son atendidas las carreras representado por el 61%, a diferencia del sistema Taxiphone respondieron 53 usuario que siempre son atendidas las carreras representado por el 83%, datos que se encuentran ilustrados en la figura Nº V.139

TABLA NºV.28. Tiempo de espera del usuario

TIEMPO	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
1 – 5 minutos	11	17	32	50
6 – 10 minutos	16	25	17	26
11 – 15 minutos	24	38	11	17
Más	13	20	4	6
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

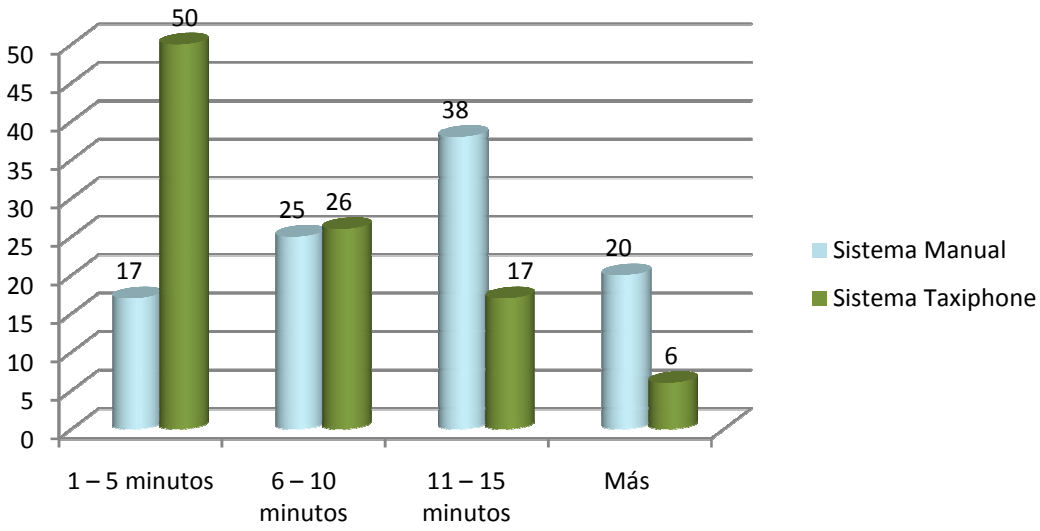


Figura Nº V. 140. Tiempo de espera del usuario

Análisis:

De acuerdo a los datos obtenidos en las encuestas se los presenta en la tabla Nº V. 28, en donde se realiza la comparación entre los sistemas manual y Taxipone, se obtuvo que el tiempo de espera por parte de los usuarios para la carrera en el sistema manual es de 11 a 15 minutos representados por la respuesta de 24 usuarios que representa el 38 %, a diferencia que con el sistema Taxiphone 32 usuarios respondieron que el tiempo de espera es de 1 a 5 minutos los que se encuentran representados por el 50%, dichos datos se encuentran plasmados en la figura Nº V.140

TABLA Nº V.29. Evaluación de atención del servicio de la cooperativa de Taxi

EVALUACIÓN	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Excelente	17	27	21	33
Muy Buena	15	23	18	27

Buena	13	20	12	19
Regular	11	17	9	14
Insuficiente	5	9	3	5
Pésimo	3	4	1	2
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

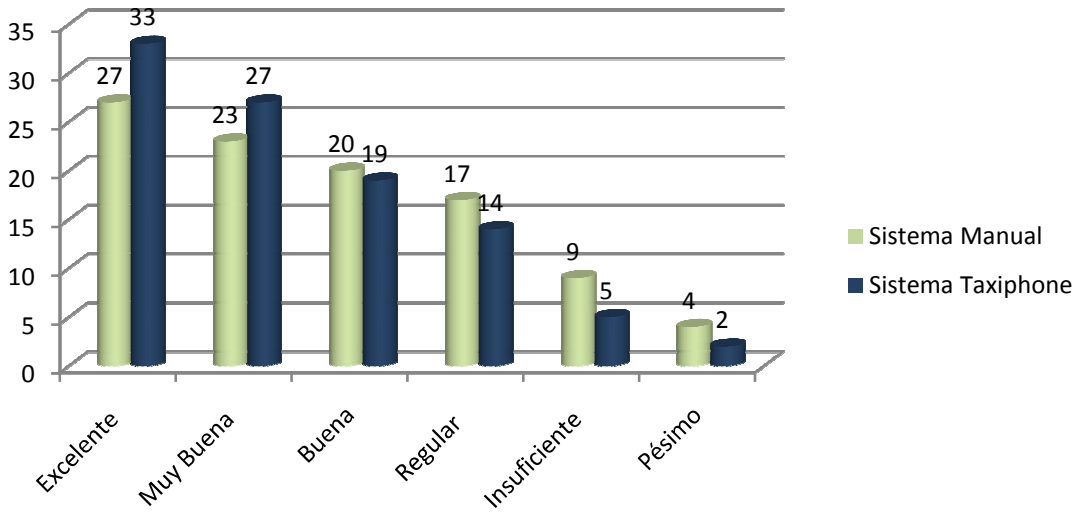


Figura Nº V. 141. Evaluación de atención del servicio de la cooperativa de Taxi

Análisis:

Los datos de la Tabla Nº V.30, representan la opinión de los usuarios en cuanto a la atención que brinda la cooperativa, con ésta información se realizó la comparación entre el sistema manual y el Taxiphone, en donde 17 personas respondieron que la atención del servicio de la cooperativa con el sistema manual es excelente con un porcentaje del 27%, de manera similar para la atención del servicio con el sistema Taxiphone que 21 usuarios opinaron que es excelente representado por el 33%, datos que se encuentran representados en la figura Nº V. 141

TABLA NºV.30. Al solicitar el taxi se traslada con seguridad al lugar de destino

SEGURIDAD	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Si	39	61	54	84
No	25	39	10	16
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

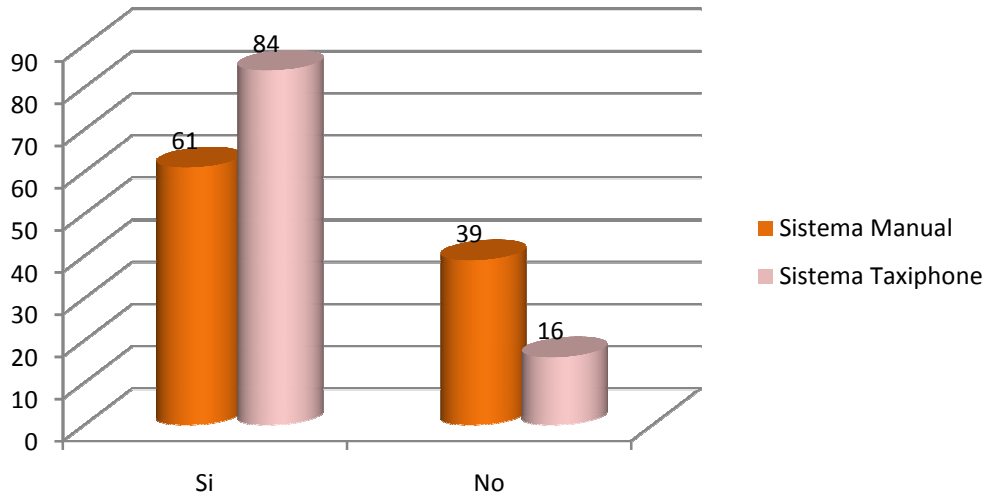


Figura Nº V. 142. Al solicitar el taxi se traslada con seguridad al lugar de destino

Análisis:

La opinión de los usuarios se encuentra representada en la tabla Nº V.30, indicando la seguridad para trasladarse a su lugar de destino utilizando el taxi, realizando la comparación entre la seguridad brindada empleando el sistema manual y taxiphone, respondieron 39 usuarios que si se trasladan seguras utilizando el servicio del sistema manual representado por el 61%, de manera casi similar opinaron de la seguridad trabajando con el sistema Taxiphone en donde respondieron 54 personas que prefieren solicitar el servicio del taxi para trasladarse con seguridad a su destino, dato que representa el 84% presentado en la figura Nº V.142.

TABLA Nº V.31. Mantener Servicio de la Cooperativa de Taxi

SERVICIO	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Si	35	55	62	97
No	29	45	2	3
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

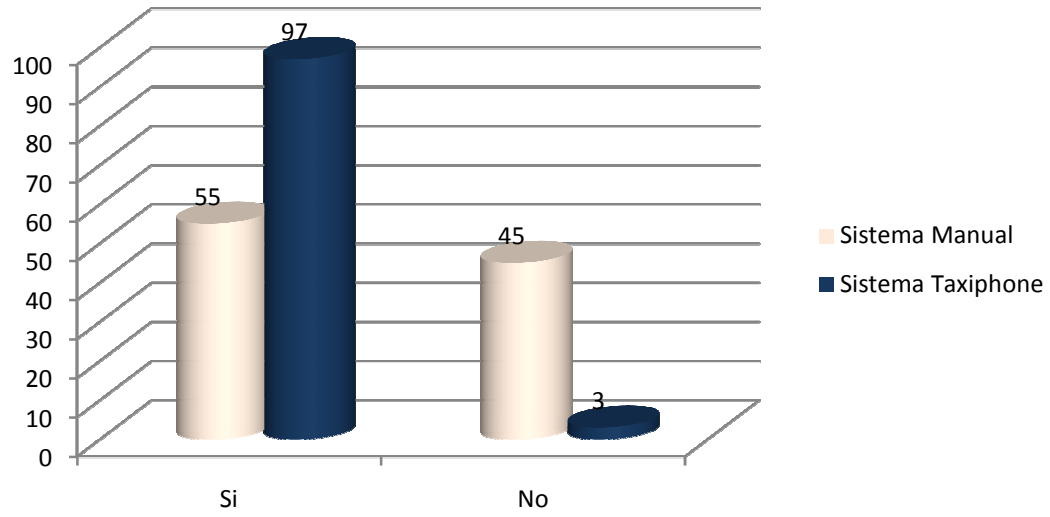


Figura Nº V. 143. Mantener el Servicio de la Cooperativa de Taxi

Análisis:

Según la opinión de los usuarios que se presentan en la tabla Nº V.31, se realiza la comparación entre los sistemas manual y taxiphone con respecto a mantener el servicio de la cooperativa, en donde 35 usuarios respondieron que si se mantendría con el servicio de la cooperativa representado por el 55%, de manera semejante para el servicio con el sistema Taxiphone respondieron 62 usuarios que si se mantendrían con el servicio representado por el 97%, datos que se obtuvo de la figura Nº V.143

TABLA Nº V.32. Mejoramiento del servicio de la Cooperativa

SERVICIO	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Organización entre taxistas	17	27	21	33
Seguridad a usuarios	15	23	18	27
Rapidez en la atención	13	20	12	19
Registro de usuarios	11	17	9	14
Otros.	5	9	3	5
TOTAL	**64**	**100%**	**64**	**100%**

FUENTE.- Encuesta dirigida a usuarios de la Cooperativa Taxiphone

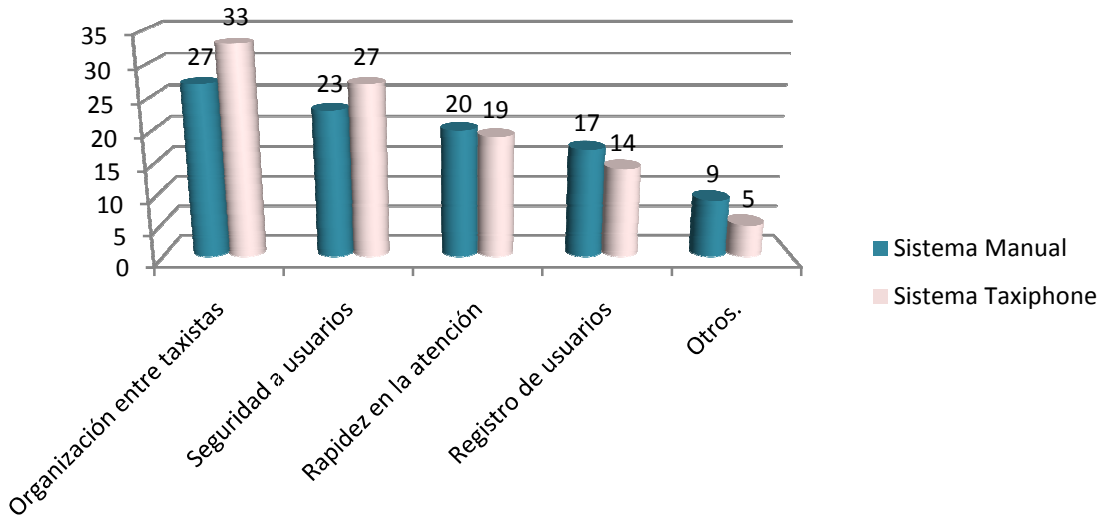


Figura Nº V. 144. Mejoramiento del servicio de la Cooperativa

Análisis:

De acuerdo a los datos presentados en la tabla Nº V.32, se muestra la opinión para el mejoramiento de algún servicio de la cooperativa empleando el sistema manual o el sistema Taxiphone, en donde nos indican en ambas circunstancias que se deberá cambiar inicialmente la organización de los taxistas, de los cuales 17 personas opinaron para el sistema manual y 21 para el sistema Taxiphone representados por el 27% y 33% respectivamente como se muestra en la figura Nº V.144

ENCUESTA A TAXISTAS.

TABLA Nº V.33. Tiempo que pertenece a la Cooperativa Taxiphone

TIEMPO	NUMERO	PORCENTAJE
1 – 5 meses	14	40
6 a 11 meses	10	29
1 a 2 años	7	20
Más de 2 años	4	11
TOTAL	**35**	**100%**

FUENTE.- *Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba*

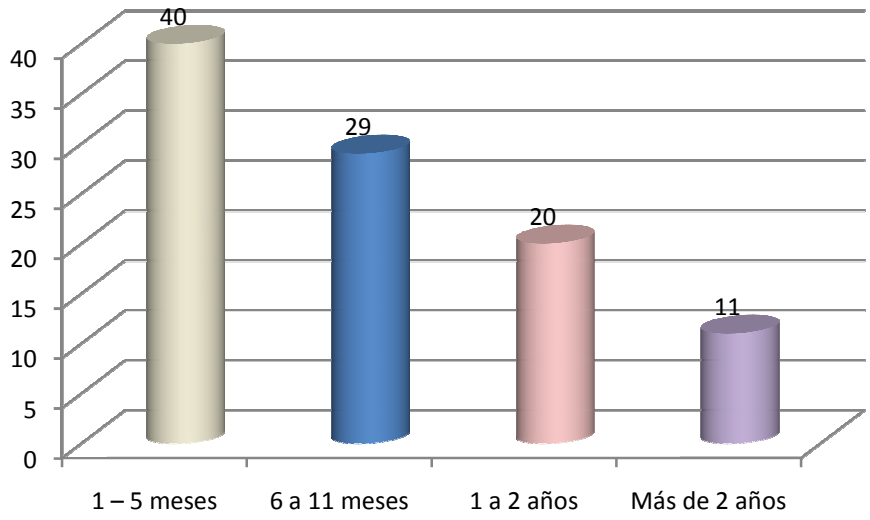


Figura Nº V. 145. Tiempo que pertenece a la Cooperativa Taxiphone

Análisis:

Al realizar la encuesta dirigida a taxista de la Cooperativa Taxiphone señalaron que el 40% trabajan de 1 a 5 meses, mientras que el 11% de ellos trabajan más de dos años, datos que se encuentran contemplados en la tabla Nº V.33 y en la figura Nº V.145

TABLA Nº V.33. Horario de Trabajo

HORARIO	Nº	PORCENTAJE
Madrugada	9	26
Mañanas	8	23
Tardes	8	23
Noche	7	20
Distintos horarios	3	9
Otros.	0	0
TOTAL	**35**	**100%**

FUENTE.- Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba

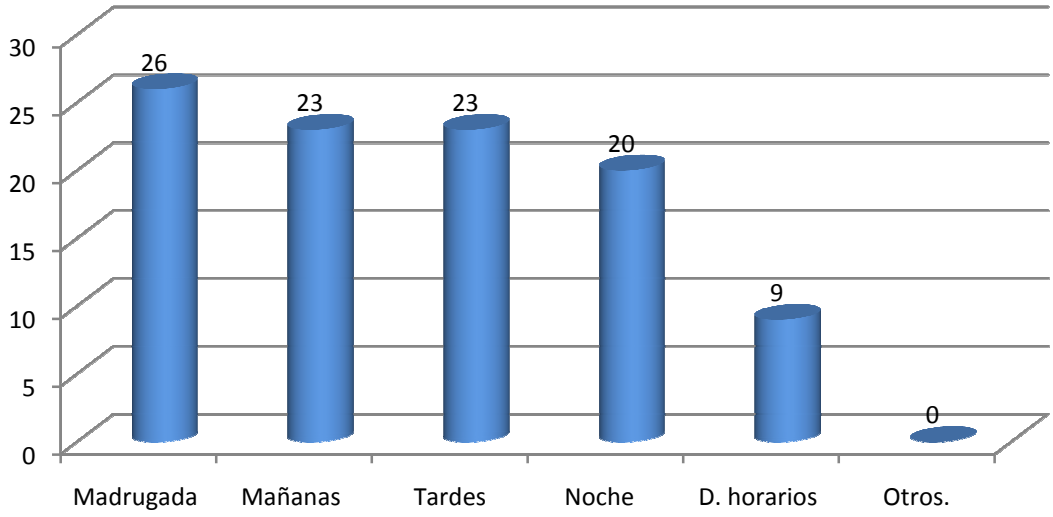


Figura Nº V. 146. Horario de Trabajo

Análisis:

De acuerdo a los datos presentados en la tabla Nº V.33 y la figura Nº V.146, se puede apreciar que el horario de trabajo de los taxistas es en la madrugada representada por el 26%, seguido del horario de la mañana y tarde con el 23%

TABLA Nº V.34. Inconvenientes para ubicar las direcciones domiciliarias de los usuarios

INCONVENIENTES	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
No proporcionas dirección exacta	10	29	7	20
No tiene dato exacto de la dirección	12	34	5	14
Los usuarios no salen al llamado del taxi	8	23	10	29
Ninguno	5	14	13	37
TOTAL	**35**	**100%**	**35**	**100%**

FUENTE.- Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba

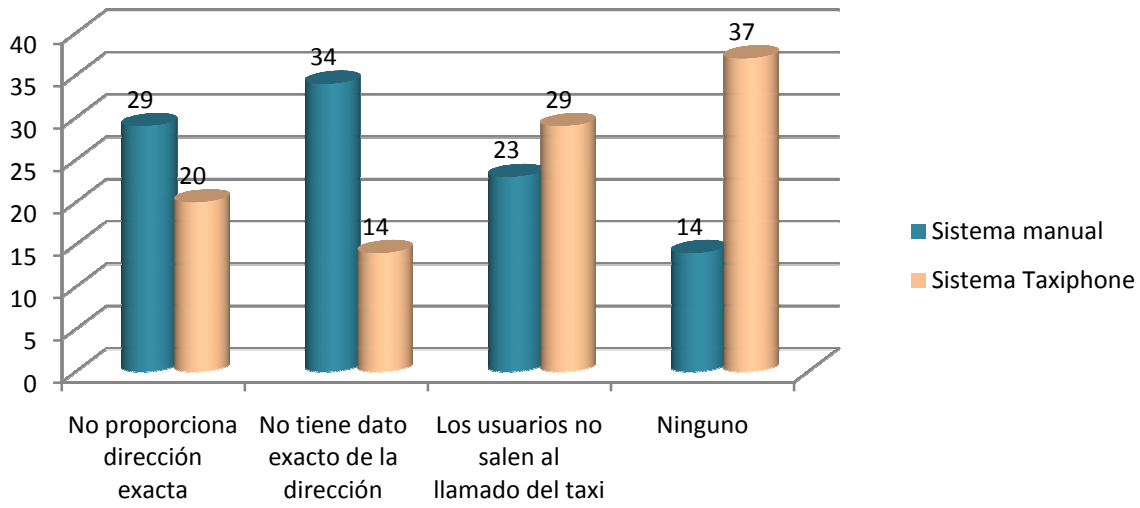


Figura Nº V. 147. Inconvenientes para ubicar las direcciones domiciliarias de los usuarios

Análisis

La tabla Nº V.34 y la figura Nº V.147, muestra la opinión de los taxistas que indican que con la utilización del sistema manual el inconveniente que se les presenta para la ubicación de la dirección de los usuarios es que no poseen el dato exacto de la misma, representada por el 34%, a diferencia del sistema taxiphone no tiene inconvenientes para la ubicación de la dirección que se encuentra representada con el 37%

TABLA Nº V.35. Calificación del Registro de Usuarios de la Cooperativa

CALIFICACIÓN	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Excelente	8	23	12	34
Muy Buena	4	11	9	26
Buena	5	14	7	20
Regular	6	17	5	14
Insuficiente	10	29	2	6
Pésimo	2	6	0	0
TOTAL	**35**	**100%**	**35**	**100%**

FUENTE.- *Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba*

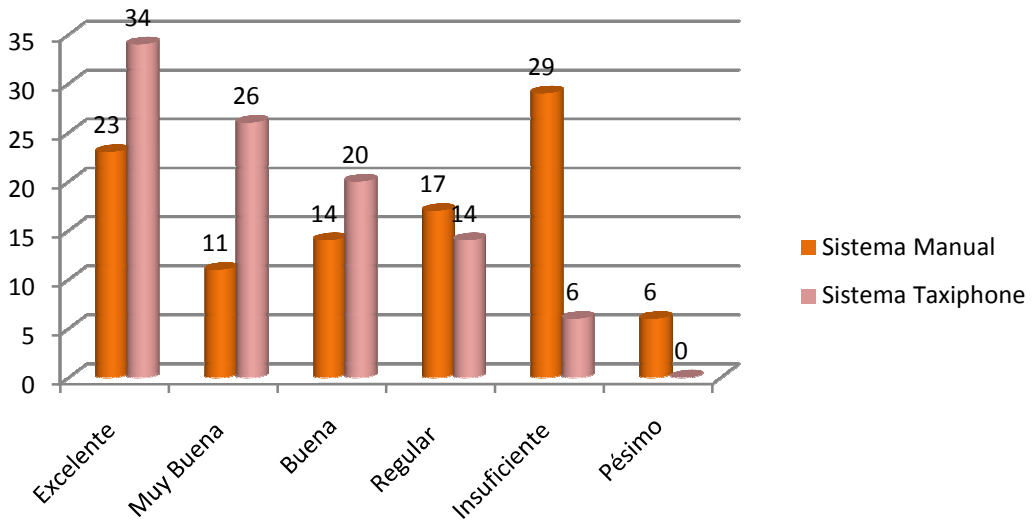


Figura Nº V. 148. Calificación del Registro de Usuarios de la Cooperativa

Análisis:

Al realizar el análisis comparativo entre los dos sistemas se pudo comprobar que el la calificación del registro de usuarios con el sistema manual es de insuficiente con el 29%, en cambio la calificación para el sistema Taxiphone es de excelente con el 34%, datos que son presentados en la tabla Nº V.35 y en la figura Nº V.148

TABLA Nº V.36. Conformidad en trabajar con los procesos actuales

CONFORMIDAD	SISTEMA MANUAL		SISTEMA TAXIPHONE	
	Nº	%	Nº	%
Si	7	20	32	91
No	28	80	3	9
TOTAL	**35**	**100%**	**35**	**100%**

FUENTE.- Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba

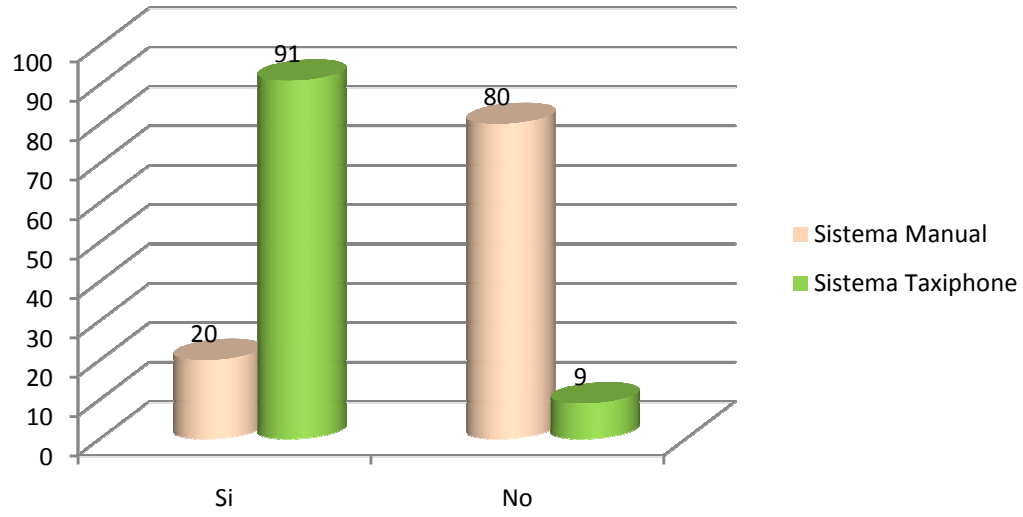


Figura Nº V. 149. Conformidad en trabajar con los procesos actuales

Análisis

Como se indica en la tabla Nº V.36 y en la figura Nº V.149, se aprecia que los taxistas no se encuentran conformes con el trabajo actual representado por el 80%, a diferencia del sistema Taxiphone que se encuentran conformes con trabajar con el sistema representado por el 91% de ellos

TABLA Nº V.37. Necesidades de la Cooperativa para mejorar su funcionamiento

NECESIDADES	NUMERO	PORCENTAJE
Cambiar a un sistema informático	28	80
Cambio de secretaria	4	11
Organización entre directiva y socios	2	6
Otros	1	3
TOTAL	**35**	**100%**

FUENTE.- *Encuesta dirigida a taxistas de la Cooperativa Taxiphone de la Ciudad de Riobamba*

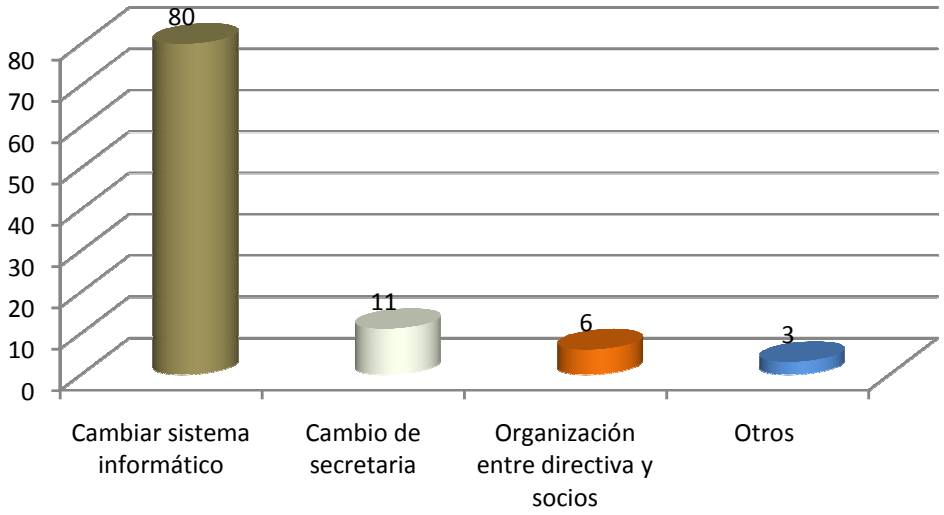


Figura Nº V. 150. Necesidades de la Cooperativa para mejorar su funcionamiento

Análisis

De acuerdo a los datos obtenidos y que se encuentran representados en la tabla Nº V.37 y la figura Nº V.50, opinan que la necesidad de la cooperativa para mejorar el funcionamiento de las actividades que se lo realizan manualmente sería cambiar esas actividades utilizando una sistema informático representado por el 80%

ENTREVISTA A LA SECRETARIA

De acuerdo a las preguntas planteadas a la secretaria se obtuvo las siguientes respuestas

¿Qué tiempo se encuentra trabajando en la Cooperativa Taxiphone?

6 meses

¿Cuál es su horario de trabajo?

En la mañana y tarde de 6 a.m a 6 p.m

¿Tienen el registro de los taxistas perteneciente a la Cooperativa?

Si, tenemos archivadas en las carpetas de los socios en el que consta la fecha de ingreso a la cooperativa

¿Cómo logra ubicar a los taxis que se encuentran trabajando y cuáles no?

Por las llamadas a la radio, pero ellos se deben reportar el momento que ingresan a laborar y de la misma manera deben notificar cuando ya salen del horario de trabajo.

¿Cada taxi tiene su horario de trabajo?

Algunos taxistas tienen sus horarios de trabajo, en cambio otros trabajan de acuerdo a la disponibilidad de tiempo, o de acuerdo a las solicitudes de sus clientes

¿Cómo logra asignar la carrera en un taxi que se encuentre disponible?

Todos los taxis que se encuentran laborando, deben reportar si se encuentran realizando alguna carrera o no, o a su vez se les llama a cada uno de los taxis para que realicen la carrera

¿Todas las llamadas recibidas son atendidas?

Todas las llamadas son atendidas cuando se encuentran dentro del horario de trabajo de la secretaria, pero algunas llamadas no se les puede asignar a tiempo los taxis para que realicen la carrera respectiva

¿Cree ud que los señores taxistas tienen problemas para la ubicación de la direcciones de los usuarios?

Claro que se les presenta inconvenientes, porque no se conoce detalladamente la ubicación del domicilio del usuario, o a su vez el usuario no proporciona la información necesaria para la ubicación

¿Cómo almacenan la información para saber realmente cuántas llamadas son atendidas?

Se tiene un cuaderno en donde se registra todas las llamadas recibidas, de la misma manera se tiene un listado de los clientes potenciales que realizan llamadas telefónicas solicitando el servicio de taxi

¿Qué servicio debería mejorarse en cuanto al funcionamiento de la cooperativa?

La atención a los usuarios, para tener el registro de las llamadas recibidas en donde se indique quien realiza la llamada e inmediatamente pueda ser atendida

GUÍA DE OBSERVACIÓN.

Tipo de Observación.- Campo

Propósito.- Obtener tiempos de espera del taxi por parte del usuario, con el sistema actual y con el sistema Taxiphone

Con la guía de observación se pretende establecer el tiempo promedio de espera por parte de usuario para que llegue el taxi a su pedido tanto en el sistema actual como con el sistema taxiphone

Tiempos de espera con el Sistema Actual

Tabla Nº V.38. Tiempo de espera con el sistema Actual

Nº	Tiempo (min)	Nº	Tiempo (min)	Nº	Tiempo (min)
1	8	23	6	45	8
2	9	24	7	46	6
3	7	25	5	47	7
4	10	26	7	48	9
5	12	27	9	49	6
6	13	28	5	50	7
7	11	29	6	51	10
8	9	30	10	52	9
9	10	31	8	53	8
10	7	32	7	54	7
11	6	33	7	55	6
12	9	34	6	56	12
13	8	35	10	57	10
14	10	36	7	58	5
15	9	37	8	59	7
16	9	38	8	60	6
17	8	39	9	61	9
18	10	40	7	62	8
19	6	41	8	63	7
20	9	42	10	64	8
21	7	43	9	65	-
22	8	44	10	66	-

Tiempos de espera con el Sistema Taxiphone

Tabla Nº V.39. Tiempo de espera con el sistema Taxiphone

Nº	Tiempo (min)	Nº	Tiempo (min)	Nº	Tiempo (min)
1	3	23	5	45	7
2	6	24	7	46	8
3	8	25	4	47	5
4	10	26	6	48	7
5	5	27	8	49	6
6	4	28	9	50	4
7	7	29	6	51	5
8	6	30	7	52	6
9	7	31	8	53	8
10	5	32	8	54	5
11	4	33	9	55	6
12	6	34	8	56	9

13	3
14	5
15	6
16	8
17	9
18	8
19	9
20	9
21	7
22	6

35	7
36	6
37	7
38	9
39	6
40	9
41	8
42	6
43	3
44	11

57	8
58	4
59	5
60	9
61	5
62	6
63	9
64	8
65	-
66	-

Teniendo los datos obtenidos del sistema actual como del sistema Taxiphone presentados en las Tablas Nº V.38 y V.39, respectivamente, cabe indicar que fueron tomados aleatoriamente en distintos días y a distintos usuarios, se procederá a sacar la media de cada una de ellas para establecer tiempos promedios en cada sistema empleado.

La media del sistema Actual es:

$$\bar{x} = \frac{519}{64} = 8.11\ minutos$$

La media del sistema Taxiphone es:

$$\bar{x} = \frac{420}{64} = 6.56\ minutos$$

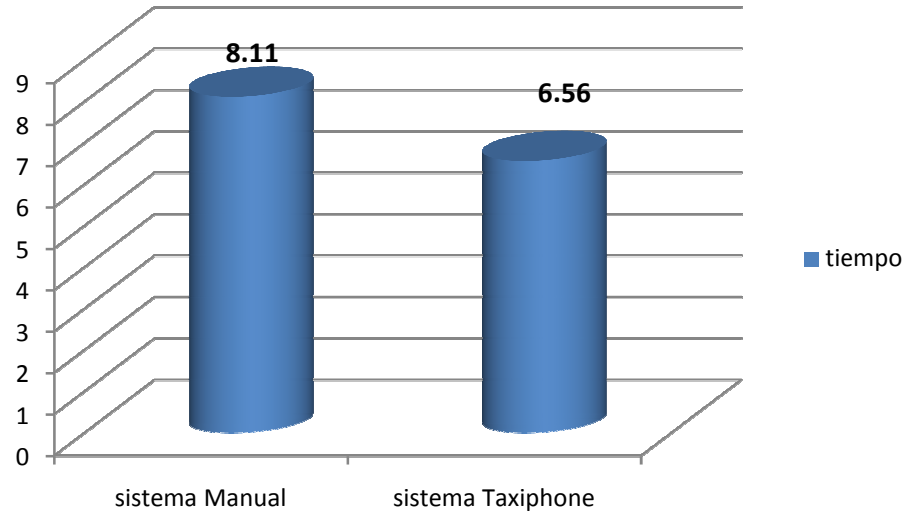


Figura Nº V. 151. Tiempos promedios de espera

Análisis

De acuerdo a las frecuencias tomadas en las tablas Nº V.38 y V.39, se calculo la media aritmética entre ellas, en donde la media de los tiempos de espera utilizando el sistema actual es de 8,11 minutos, en cambio que la media de los tiempo de espera con el sistema taxiphone es de 6.56 minutos como se muestra en la figura Nº V.151, consiguiendo de esta manera que con el sistema Taxiphone se genera rapidez en la atención de las llamadas de los usuarios

CONCLUSIONES

1. Profundizar el estudio de los ORM nos sirvió para establecer la relación directa que existe entre el mapeo objeto relacional con el framework de persistencia NHibernate, ya que los dos van de la mano en el momento que se desarrollan aplicaciones bajo estas características, en las que se debe tener en cuenta el proceso para obtener la estructura relacional correcta de la aplicación a desarrollar.
2. De acuerdo al estudio secuencial y cronológico del framework NHibernate, se puede apreciar la amplitud y el auge que está tomando éste framework para las multiplex ventajas que ofrece al desarrollador, basta con partir del diagrama de la base de datos para conseguir la estructura de la aplicación, por tanto se puede aplicar el framework para aplicaciones Windows, web o móviles sin perder el concepto real de la aplicación a desarrollarse.
3. Para conocerle más al framework NHibernate se estableció el patrón de diseño el mismo que proporciona los pasos como guía para el correcto uso y configuración, que servirá como apoyo para la creación de nuevas aplicaciones
4. Con la implementación del framework NHibernate, se generó el modelo ORM plasmado desde el diseño de la base de datos, que básicamente se centra en tomar como punto de partida el modelo relacional de la base de datos
5. Se desarrolló el sistema Taxiphone que brinda la automatización de las actividades de la cooperativa Taxiphone, en la que se utilizó el framework NHibernate, ORM y archivos de mapeo para conseguir el sistema para dicha cooperativa

RECOMENDACIONES

1. Con el estudio de los ORM, se debe tener en cuenta que los conceptos framework y ORM son conceptos diferentes aunque se posean el mismo eje de desarrollo, pues los dos van de la mano para conseguir las aplicaciones deseadas, por tanto hay que primeramente diferenciar las acciones que cada uno de ellos realiza
2. Hay que considerar que cada nueva versión que surja de NHibernate va hacer para proporcionar mejoras en el desarrollo pero que además hay que conocer el motivo por la que han cambiando o han surgido nuevos componentes para NHibernate
3. El patrón de diseño planteado en éste documento se lo realizó para una aplicación Windows de 3 capas, este patrón podría variar para aplicaciones web o Windows, ya que aquí se necesitaría emplear nuevos componentes de NHibernate como el Burrow, Helper entre otros
4. Para la implementación del framework NHibernate, se recomienda revisar ésta documentación ya que será de gran ayuda conocer las librerías y parámetros que se necesitan para su configuración y mapeo
5. El desarrollo del sistema Taxiphone se realizó en base a la configuración de NHibernate, que se necesita tener la ubicación del archivo de configuración en el cadena de conexión para que la aplicación reconozca dicha configuración de los contrario se perderá la comunicación con la base de datos.

RESUMEN

El objetivo de la investigación, realizar el estudio de framework de persistencia NHibernate basado en la tecnología Java y aplicarle en la tecnología .NET desarrollando un sistema para la detección de llamadas en la Cooperativa de taxis Taxiphone, del Cantón Riobamba, Provincia de Chimborazo.

Se utilizo la metodología Microsoft Solution Framework MSF para el desarrollo del sistema de detección de llamadas telefónicas en el lenguaje de programación C#.Net de Visual Studio 2005 empleando el framework de persistencia NHibernate

Se logró establecer la configuración de archivos de conexión con la base de datos, así también se creó los archivos para la capa de acceso a datos bajo el framework NHibernate, consiguiendo ingresar, abstraer y manipular datos desde la capa de presentación hacia la base de datos de una manera segura y rápida, facilitando al desarrollador la creación de excesiva líneas de código parea lograr un propósito.

Con la implantación del sistema logró determinar que la automatización de las actividades en la cooperativa mejoró la productividad con la eliminación de errores, reducción de tiempo, buena atención y seguridad a sus usuarios a diferencia del sistema manual con el que se desempeñaban

Se recomienda experimentar nuevas herramientas de desarrollo como el framework NHibernate que brinda facilidades y ventajas en el ámbito de la informática; A las cooperativas de taxis, se recomienda adquirir el sistema “Taxiphone” que les brindará mejoras con la automatización en el desempeño de sus actividades y conseguir la información que se desea de manera oportuna y veraz.

SUMMARY

The objective of the research work is the carry out the NHibernate persistence framework study, based on Java technology in order to apply it in ":NET" technology, by developing a system for call detection in "taxi phone", taxi company, from Riobamba canton, Chimborazo Province.

The Microsoft Solution Framework (MSF) methodology was used for the call detection system development in the C#.NET programming language C#.NET of Visual Studio 2005, by using the NHibernate persistence framework.

It was possible to establish the file connection configuration with the database, as well as the files for the access layer to data under the framework NHibernate were created, through this, it was possible to enter, abstract and manipulate data from the presentation layer towards the database in a safe and fast way, making it possible for the developer the creation of excessive cod lines in order to achieve a purpose.

With the installation of the system, it was possible to determine that the activities automation in the taxi company has improved the productivity with mistakes elimination, time reduction, good attention and security to its users in contrast to the manual system with the one they worked

It is recommended to use new development tools, such as the NHibernate framework, which offers facilities and advantages in the computer science area; it is recommended for the taxi company to acquire the "Taxiphone" system, which will offer improvements with the automation in the development of the activities, in order to get the required information in a opportune and truthful way

GLOSARIO

BASE DE DATOS

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso que ofrece un amplio rango de soluciones al problema de almacenar datos.

CAPA DE ACCESO A DATOS

Capa que sirve entre como puente entre la capa lógica de negocio y el proveedor de datos. Este capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa.

CAPA DE PRESENTACIÓN

Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable entendible y fácil de usar para el usuario.

CAPAS DE NEGOCIO

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso porque es aquí donde se establecen todas las reglas que deben cumplirse., ésta capa se comunica con la

capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

FRAMEWORK

Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar, en el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.

MAPEO OBJETO RELACIONA

El mapeo objeto-relacional, más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

PERSISTENCIA

Se entiende por persistencia como la acción de preservar la información de un objeto de forma permanente, pero a su vez también se refiere a poder recuperar la información del mismo para que pueda ser nuevamente utilizada

ANEXO

ANEXO Nº 1

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS

ENCUESTA A USUARIOS

OBJETIVO.- Conocer la calidad de servicio en tiempo y seguridad que brinda la Cooperativa de Taxis TAXIPHONE a todos sus usuarios de la ciudad de Riobamba

1. ¿En qué zona de Riobamba habita?

- ☐ Norte
- ☐ Sur
- ☐ Este
- ☐ Oeste

2. ¿Con qué frecuencia solicita el servicio de taxis de nuestra cooperativa?

- ☐ Siempre
- ☐ Frecuentemente
- ☐ Rara vez
- ☐ Nunca

3. ¿Cada vez que solicita el taxi siempre es atendido en su carrera?

- ☐ Si
- ☐ No

4. El tiempo promedio de espera desde la llamada hasta la llegada del taxi es:

- ☐ 1 – 5 minutos
- ☐ 6 – 10 minutos
- ☐ 11 – 15 minutos
- ☐ Más

5. ¿Cómo evalúa la atención del servicio de la cooperativa de taxis?

- ☐ Excelente
- ☐ Muy Buena
- ☐ Buena
- ☐ Regular
- ☐ Insuficiente
- ☐ Pésimo

6. Cree usted que solicitar un taxi se traslada con seguridad al lugar de destino?

- ☐ Si
- ☐ No

¿Por qué?

_____

7. ¿Mantendría el servicio de nuestra cooperativa?

- ☐ Si
- ☐ No

¿Por qué?

_____

8. Piensa que debería mejorar el servicio de la cooperativa, en cuanto a:

- ☐ Organización entre taxistas
- ☐ Seguridad a usuarios
- ☐ Rapidez en la atención
- ☐ Registro de usuarios
- ☐ Otros

Cuáles

_____

GRACIAS POR SU COLABORACIÓN

ANEXO Nº 2

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS

ENCUESTA A TAXISTAS

OBJETIVO.- Analizar problemas existentes dentro de la Cooperativa con el control del trabajo de los taxistas

1. ¿Qué tiempo pertenece a la Cooperativa Taxiphone?

- ☐ 1 a 5 meses
- ☐ 6 a 11 meses
- ☐ 1 a 2 años
- ☐ Más de 2 años.

2. ¿Cuál es su horario de trabajo?

- ☐ Solo en la madrugada
- ☐ Solo en la mañana
- ☐ Solo las tardes
- ☐ Solo las noches
- ☐ En distintos horarios
- ☐ Otros

3. ¿qué inconvenientes se le presenta para ubicar a tiempo las direcciones domiciliarias de los usuarios?

- ☐ No proporcionan la dirección exacta
- ☐ No se tiene un dato exacto de la dirección
- ☐ Los usuarios no salen ante el llamado del taxi
- ☐ Ninguno

4. ¿Cómo califica el sistema de Registro de Usuarios de la Cooperativa?

- ☐ Excelente
- ☐ Muy Buena
- ☐ Buena
- ☐ Regular
- ☐ Insuficiente
- ☐ Pésimo

5. Se siente a gusto trabajar con el nuevo sistema taxiphone?

- ☐ Si
- ☐ No

¿Por qué?

_____
_____

6. ¿Qué piensa que necesita la Cooperativa para mejorar su funcionamiento?

- ☐ Cambiar a un sistema informático
- ☐ Cambiar de secretaria
- ☐ Organizarse entre la directiva y socios
- ☐ Otros_____

GRACIAS POR SU COLABORACIÓN

ANEXO Nº 3

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS

ENTREVISTA A LA SECRETARIA

OBJETIVO.- Analizar problemas existentes en las actividades de la Secretaría de la Cooperativa.

1. ¿Qué tiempo se encuentra trabajando en la Cooperativa Taxiphone?
   _____
2. ¿Cuál es su horario de trabajo?
   _____
3. ¿Tienen el registro de los taxistas perteneciente a la Cooperativa?
   _____
   _____
4. ¿Cómo logra ubicar a los taxis que se encuentran trabajando y cuáles no?
   _____
   _____
5. ¿Cada taxi tiene su horario de trabajo?
   _____
   _____
6. ¿Cómo logra asignar la carrera en un taxi que se encuentre disponible?
   _____
   _____
   _____
7. ¿Todas las llamadas recibidas son atendidas?
   _____
8. ¿Cree ud que los señores taxistas tienen problemas para la ubicación de la direcciones de los usuarios?
   _____
   _____
9. ¿Cómo almacenan la información para saber realmente cuántas llamadas son atendidas?
   _____
   _____
10. ¿Qué servicio debería mejorarse en cuanto al funcionamiento de la cooperativa?

    _____

GRACIAS POR SU COLABORACIÓN

ANEXO Nº 4.

MANUAL DE USUARIO

1. INTRODUCCIÓN

Bienvenidos al Manual Usuario del sistema "Taxiphone"; se describirá el funcionamiento de la aplicación, el cual proporcionara al usuario facilidad al encontrar detallado cada uno de los pasos a seguir para realizar el manejo y ejecución de la aplicación en forma correcta.

Entre las opciones constará lo siguiente:

- ✓ Instalación del sistema Taxiphone
- ✓ Ayuda de las opciones del sistema

La presente aplicación está destinada a todos los socios. Secretarias y dirigentes de la cooperativa Taxiphone de la ciudad de Riobamba, para que tengan una guía y sea un soporte para el correcto manejo del siste

2. GENERALIDADES DEL SISTEMA

2.1. DEFINICIONES

CD: Disco Compacto

Interfaz de usuario: Es la aplicación o presentación final que interactúa con el usuario final del sistema, esta debe ser comprensible.

Base de Datos.- Es el componente o archivo en donde se encuentra toda la información que se manipulará con el sistema Taxiphone

2.2. INSTALACIÓN

2.2.1. Requisito mínimo indispensable

Hardware

- ✓ Ordenador compatible con PC, velocidad de la CPU de 200 MHz o superior
- ✓ Memoria:256 MB
- ✓ Tarjeta de fax modem
- ✓ Monitor
- ✓ Teclado
- ✓ Mouse
- ✓ impresora

Software

- ✓ Windows 98 SE, Me, 2000 y XP
- ✓ SQL Server 2005

2.2.2. Introducción (como Ejecutar)

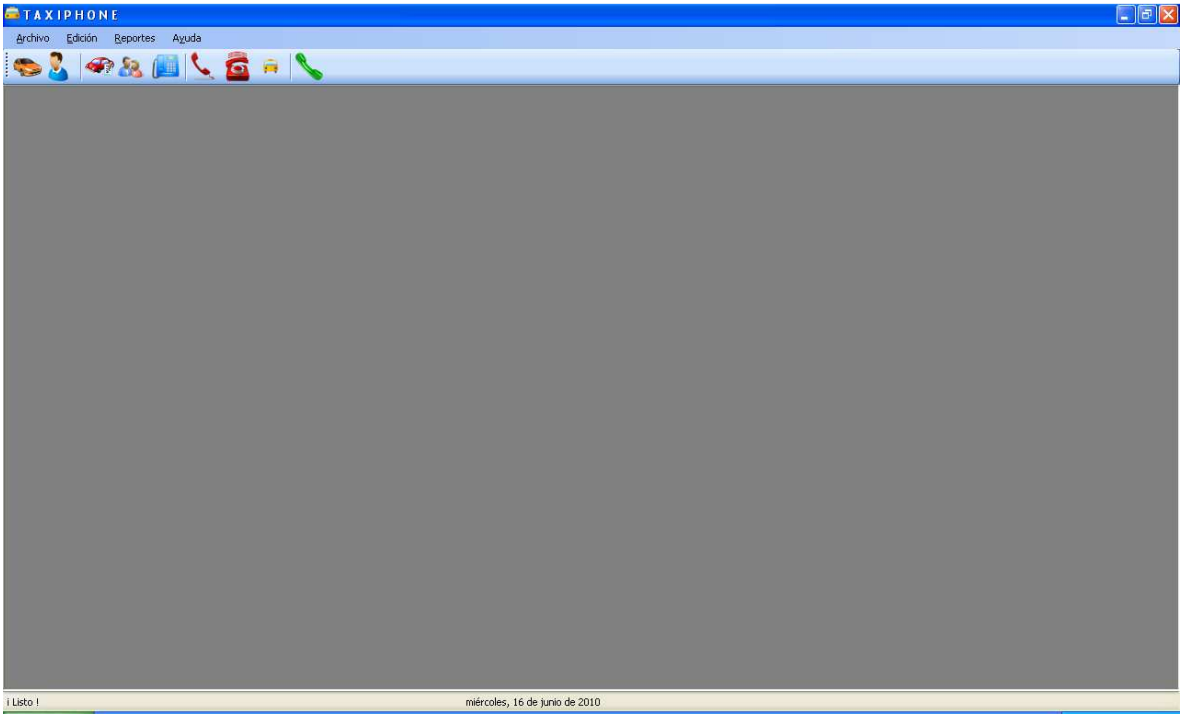
- ✓ Insertar el CD en la unidad CD-ROM de la computadora, en este CD también se encuentra el Manual de Usuario.
- ✓ Después de insertar se procederá a abrir la carpeta con el nombre de Aplicación.
- ✓ Ejecutar el archivo.

3. AYUDA DEL SISTEMA TAXIPHONE

Se presentara como pantalla incial, la que se presenta a continuación la misma que contiene un informativo del sistema

Después se presenta la pantalla de autenticación de usuarios en la que se debe ingresar el nombre y contraseña, luego se presiona el botón aceptar

Una vez autenticados como usuarios, se presenta la pantalla principal en donde se encuentra el menú con la barra de herramienta para acceder a las distintas opciones del sistema



Una de las opciones es para insertar a los usuarios del sistema, para lo cual se presenta la pantalla como a continuación se muestra

La siguiente pantalla se utiliza para insertar a un nuevo taxi, con toda la información del taxi, una vez ingresados se presiona el botón aceptar, en caso no se requiere guardar, se presiona el botón cancelar

También se puede cambiar el estado del taxi, en la pantalla que se presenta a continuación, para lo cual primero debemos presionar el botón nuevo y se nos activa los elementos para realizar la búsqueda del taxi que se desea cambiar el estado, se presiona el botón buscar y se presenta la información del taxi, modificamos el estado y presionamos el botón guardar

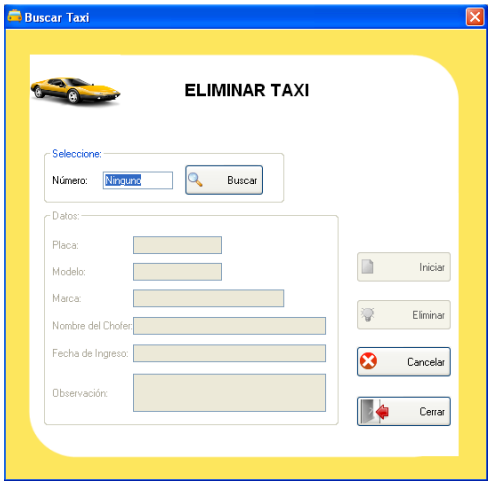
Para asignar una carrera se nos presenta la siguiente pantalla, se presiona el botón Nuevo para que se active los elementos, aquí se ingresará los datos de la carrera, cuando se termina de ingresar los datos se presiona el botón asignar

Para cambiar la disponibilidad del taxi, se nos presentará la pantalla como se muestra, en la que se presiona el botón Nuevo y se ingresa el número del taxi a buscar para lo cual se presiona el botón Buscar para que se nos muestre la información del taxi, se presiona el botón disponible y se cambia la disponibilidad del taxi

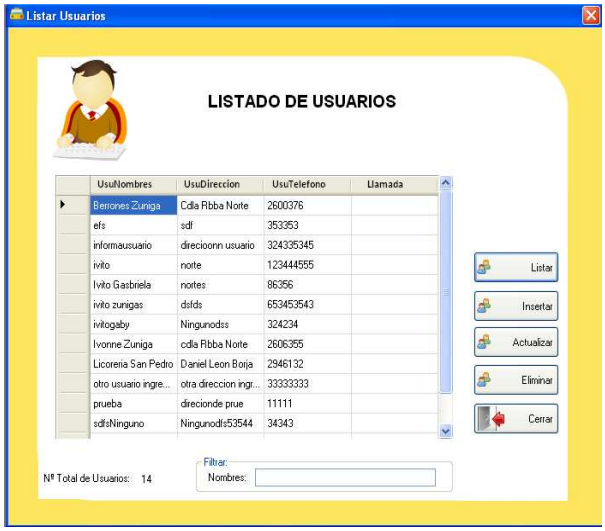
Cuando se vaya a conestar alguna llamada telefónica se presentará la pantalla como se muestra, y se debe pulsar el botón contestar para que se muestre la información del usuario en la pantalla, para luego asignarle la carrera con el botón Asignar

Para modificar la información del usuario se presenta la siguiente pantalla, para lo que es necesario de antemano realizar la búsqueda del usuario, se debe presionar el botón nuevo para que se active los elementos de búsquedas, una vez que se ha ingresado el nombre del usuario, se presiona el botón buscar mostrándonos la infomación del usuario, cuando se haya modificado se pulsa el botón modificar

Para eliminar a un usuario se sigue los mismos pasos con que se actualizaron



Para listar la información se presenta la siguiente pantalla en la que además se podrá realizar filtros de búsquedas, si se desea imprimir el reporte, se presiona el botón vista preliminar



BIBLIOGRAFÍA

BIBLIOGRAFÍA DE INTERNET

NHIBERNATE

http://jmarcost.blogspot.com/2008/02/nhibernate-intellisense-para-archivos.html

2009-08-10

http://devsoftx.wordpress.com/2008/06/04/arquitectura-de-nhibernate-hibernate-for-net/

2009-08-10

http://www.hibernate.org/

2009-08-10

http://www.forosdelweb.com/f29/nhibernate-c-2-0-preguntas-inicio-542374/

2009-06-16

http://devsoftx.wordpress.com/2008/05/30/nhibernate-modelo-de-persistencia-de-datos-en-net/

2009-06-16

http://www.codeproject.com/KB/dotnet/nhibernatept1.aspx

2009-06-16

http://www.theserverside.net/tt/articles/showarticle.tss?id=NHibernate

2009-06-16

http://www.theserverside.net/blogs/thread.tss?thread_id=42261

2009-06-16

http://blogs.hibernatingrhinos.com/nhibernate/Default.aspx

2009-06-16

http://blogs.hibernatingrhinos.com/nhibernate/archive/2008/04/01/your-first-nhibernate-based-application.aspx

2009-06-16

http://blog.jaimecg.com/2006/10/ejemplo-simple-de-nhibernate-en-mono.html

2009-06-16

http://www.mail-archive.com/nhibernate-hispano@googlegroups.com/

2009-06-16

http://www.codersource.net/published/view/280/nhibernate_part_iii.aspx

2009-06-16

FRAMEWORK

http://www.alegsa.com.ar/Dic/framework.php

2009-08-25

http://es.wikipedia.org/wiki/Framework

2009-08-25

http://www.alegsa.com.ar/Dic/.net%20framework.php

2009-08-25

http://www.webestilo.com/aspnet/aspnet00.phtml

2010-03-02

http://jordisan.net/blog/2006/que-es-un-framework/

2010-03-02

http://www.devjoker.com/contenidos/Conceptos-generales-NET/122/-NET-FrameWork.aspx

2010-03-02

http://social.msdn.microsoft.com/Forums/es-ES/generales/thread/49c5315b-31e6-45bc-82b2-0749cf82bff1

2010-03-02

ORM

http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

2009-11-30

http://www.rumorismo.com/pr/definicion-de-orm-online-reputation-management/

2009-11-30

http://www.librosweb.es/symfony_1_1/capitulo8/por_que_utilizar_un_orm_y_una_capa_de_abstraccion.html

2009-11-30

http://es.wikipedia.org/wiki/.NET

2009-11-30

http://www.librosweb.es/symfony_1_0/capitulo1/conceptos_basicos.html

2009-12-08

http://www.softwarelibre.net/%C2%BFsiguen_siendo_las_bases_de_datos_relacionales_la_mejor_opci%C3%B3n

2009-12-08

http://devsoftx.wordpress.com/2008/12/27/nhibernate-aplicando-patrones-de-disenio-a-una-aplicacion-que-utiliza-nhibernate-parte-1/

2009-12-08

IDENTIFICADOR DE LLAMADAS

http://www.ecualug.org/2002/08/18/faq/9_internet_modems/9_1_que_es_un_modem_y_un_puerto_serie

2010-02-21

http://www.juntadeandalucia.es/averroes/recursos_informaticos/curso_internet/curso/conmodem.htm

2010-01-13

http://www.wikilearning.com/tutorial/ppp/20625-9

2010-01-13

http://www.lawebdelprogramador.com/preguntas/vercontestada.php?id=227&texto=C+sharp&pagina=1

2010-01-13

http://blogdediegoramirez.blogspot.com/2008/05/creando-un-identificador-de-llamadas.html

2010-01-13

http://www.forosdelweb.com/f29/modem-con-identificador-llamadas-caller-id-con-c-694795/

2010-02-21