UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

INSTITUTO SUPERIOR POLITÉCNICO "JOSÉ ANTONIO ECHEVERRÍA"

FACULTAD DE INGENIERÍA INDUSTRIAL

CENTRO DE ESTUDIOS DE INGENIERÍA Y SISTEMA

INGENIERÍA EN INFORMÁTICA

Sistema de Reservación de Pase Masivo

Trabajo para optar por el Título de

Ingeniería en Informática

AUTOR

Liecel Hernández Sotolongo

TUTOR

Ing. Rudel Cárdenas Díaz

Ciudad de la Habana

Junio 2005

Agradecimientos

… A mi familia, en especial mis padres y hermano, que siempre me han apoyado y ayudado en todo y a pesar de todo…
A mis amigos que no menciono, perdónenme pero todos saben quienes son y son demasiados para una sola página.
A los que están y los que no, a los que dieron por su ayuda y a los que no dieron por hacerme más fuerte…
A mi tutor y amigo Rudel, por luchar tanto y sin quien nada se habría logrado.
A Luke, por su ayuda …
Al Napo, Erick y Mabe …. Que tanto me dieron ……
A Feston .. que negro! Y Pedro … mis hermanos, Adrián, Vismar, Andy, Renier… mis otros hermanos …
A los RALPDIK, por estar siempre aunque estén lejos… incluso desde el más allá…

A Mainel, Natacha, Adnel, Sandy, al Yoni, Liana, Dayner…
Y tantos otros que ahora no recuerdo y que no caben porque el libro que han escrito en mi vida es demasiado alto para esta hoja …

A la CUJAE por enseñarme tanto…..
A la Revolución por todo lo que me ha dado y sigue dando ….

A todos … gracias por hacerme quien soy …….

A mis padres… por confiar en mí ...

Resumen

El presente proyecto consiste en la confección de una herramienta que sea capaz tanto de manipular como de generar la información necesaria para gestionar las distribuciones de viajeros para cada transporte de los pases masivos de la Universidad de las Ciencias Informáticas (UCI). La automatización de dicho proceso se hace necesaria debido al proyecto de Informatización de la UCI y a lo difícil de distribuir los viajeros de forma manual, ya que su número va en aumento.

Además se propone lograr una integración entre los diferentes servicios existentes así como un mejor aprovechamiento de la infraestructura informática del centro, garantizando de igual forma un adecuado flujo de la información y un rápido acceso a la misma por parte de los usuarios interesados.

Previamente se habían elaborado otros sistemas para estas funciones pero no satisfacían completamente los requerimientos del sistema y no funcionaban de forma óptima. Es por ello que el presente está diseñado para hacer un óptimo uso de los servicios y recursos existentes en la UCI y lograr una completa integración con los sistemas de los cuales solicita información necesaria para satisfacer los antes mencionados requerimientos.

Índice

Introducción

La Universidad de las Ciencias Informáticas es un centro que, por ser único en su tipo en todo el país, alberga en sus instalaciones a más de cinco mil personas, estas personas deben ser transportadas varias veces al año hacia sus provincias de residencia ya sea por motivos festivos o por la culminación del período docente. Para transportar todas estas personas la Universidad compra los pasajes necesarios a las Agencias de Transporte del país según la cantidad de viajeros y sus destinos y luego los distribuye entre estos garantizando así el transportamiento del personal.

Cada año ingresan a la Universidad de las Ciencias Informáticas (UCI) nuevos estudiantes, dirigentes, trabajadores, profesores y familiares de estos, es por ello y unido a la necesidad de los usuarios de conocer la información referente a sus pasajes que se hace necesaria la informatización del proceso de distribución. La UCI hace las reservaciones de los pasajes basándose en las cantidades de personas por municipios y provincias por lo que, luego de obtener los pasajes, es necesario distribuir a las personas por los distintos medios de transporte e informarles de su ubicación y fecha de partida. Todo lo anterior y el hecho de que la UCI está atravesando un proceso de completa digitalización, ha propiciado la idea de automatizar la gestión de distribuir a los pasajeros en los medios de transporte.

Comoquiera que la UCI cuenta con una infraestructura digital que proporciona los datos de las personas residentes en la institución y permite a los mismos interactuar de forma activa con dicha información, la creación de sistemas automatizados de reservación se hace tan necesaria como posible. Esto, por otra parte, posibilita tratar la información, partiendo del hecho de que se trata de datos completamente actualizados y debidamente organizados, de forma rápida y segura así como logrando cierta comodidad

para los trabajadores encargados de la manipulación y confección de la distribución.

Con anterioridad fueron concebidos diferentes sistemas que trataban de dar respuesta a estas necesidades pero solo se lograron algunas pequeñas aplicaciones que manipulaban ciertos tipos de información y daban resultados parciales por lo que la principal carga de trabajo seguía recayendo en los trabajadores encargados de la reservación.

El proceso de reservación de los pasajes es de vital importancia pues estos son los momentos en los que se estará transportando todo el personal residente en la UCI; es por ello que, conjuntamente con la gran cantidad de condiciones a tener en cuenta en la distribución, se hace necesario incluir la automatización de dicha tarea entre las prioridades del proyecto de Informatización de la Universidad. El sistema necesario para ello está dirigido a cumplir con las exigencias de la ubicación de forma completamente informatizada e integrándose a los diferentes sistemas ya implementados que proveen los datos necesarios para la realización de esta tarea. Por otra parte, esta aplicación proveerá la distribución de los usuarios en los distintos tipos de transportes, así como la información referente a la misma, la cual será generada de forma cómoda y sencilla basada en los datos y parámetros dados por los trabajadores encargados de su elaboración.

La principal tarea del sistema es la ubicación o distribución de los viajeros a transportar, sin embargo también provee todo tipo de reportes sobre esta distribución, ya sean cantidades relativas y búsqueda de personas como listas de los propios viajeros con sus respectivas ubicaciones. La aplicación ha sido diseñada tratando de darle la mayor flexibilidad posible para que permita distintas variantes al organizar una distribución, acorde con las condiciones que se necesiten y las necesidades de usuarios y trabajadores.

Debido a que es un módulo del futuro Sistema Integral de Reservaciones de la UCI, esta aplicación se ha concebido de forma tal que se integre completamente a los demás sistemas y permita la adición de nuevos submódulos con otras prestaciones que se requieran en un futuro. Para esto se realizó un profundo análisis del negocio actual de reservación con el fin de lograr un mejor entendimiento de qué se necesita, quiénes interactúan con el servicio y cómo funciona, para así poder dar una respuesta adecuada a los requerimientos del mismo teniendo en cuenta sus características y peculiaridades, sin perder la flexibilidad. También se analizaron las características de los diferentes transportes utilizados y de los boletines a emitir con el fin de lograr una ubicación rápida, eficiente y segura.

En aras de lograr una buena accesibilidad al sistema por parte de los usuarios se decidió desarrollarlo como aplicación Web, utilizando para la implementación la tecnología .NET de Microsoft.

El presente trabajo tiene como objeto de estudio toda gestión involucrada en el proceso de reservación de pasajes y distribución de los viajeros junto al diseño de la base de datos y el desarrollo de una aplicación Web multicapas; teniendo como campo de acción la Dirección de transporte de la UCI, las bases de dato del centro y los servicios Web que posibilitan el acceso a la información en las mismas. Este sistema se propone de forma general satisfacer las demandas de los usuarios del servicio de forma eficiente, rápida, segura y confortable, así como facilitar el trabajo a los encargados de la gestión. Esto será posible mediante una completa automatización del proceso, brindando una amplia gama de posibilidades tanto a los usuarios como a los trabajadores que interactúen con la aplicación y una serie de reportes que van desde obtener la ubicación dada a un usuario, hasta la lista completa de la distribución hecha por el sistema.

En función de una aplicación a la altura de las expectativas se decidió seguir una serie de pasos o tareas a desarrollar en el proceso de elaboración de la misma:

- Estudio a profundidad de todo lo relacionado con el problema en cuestión.
- Estudio de las tendencias tecnológicas actuales que podrían ayudar en la solución del problema.
- Fundamentación teórica de la aplicación a desarrollar.
- Reconocimiento de Requisitos Funcionales y no Funcionales del sistema así como de los Casos de Uso.
- Diseño de la aplicación.
- Implementación

La información contenida en este documento referente a la investigación, análisis, diseño y desarrollo de la aplicación ha sido estructurada como se describe a continuación:

Capítulo Primero: Fundamentación Teórica. Se explican los detalles del problema a resolver, analizándose la situación problémica, el campo de acción y del proceso del negocio actual.

Capítulo Segundo: Tendencias de las tecnologías actuales a tener en cuenta. Trata sobre las tendencias tecnológicas y herramientas utilizadas en el desarrollo del proyecto. Objetivos de la solución propuesta.

Capítulo Tercero: Descripción de la solución propuesta. Este capítulo aborda la descripción de la modelación del problema al que se le propone solución. Se describen los procesos del negocio, actores, modelo de objetos, diagrama de clases, requisitos y la descripción de la propuesta de sistema.

Capítulo Cuarto: Implementación de la solución propuesta. Contiene las descripciones de las clases y componentes que diseñados para la implementación de la aplicación

Capítulo Quinto: Estudio de factibilidad. Aquí se hace el estudio de factibilidad. Se analizan los resultados de Esfuerzo, Hombres por Mes, Tiempo de desarrollo y se comparan los resultados de los beneficios con el costo.

Capitulo 1

Fundamentación del Tema

1.1 - Introducción

A continuación se explica todo lo referente al problema al cual este sistema brinda solución, así como los distintos conceptos y definiciones tenidos en cuenta y utilizados en su elaboración, además de una breve descripción de los sistemas que brindan los datos necesarios y constituyen la principal fuente de información para la aplicación.

1.2 – Descripción de los principales conceptos y definiciones utilizados

En la Universidad de las Ciencias Informáticas (UCI) permanecen becados los estudiantes y una parte de los trabajadores, profesores y sus familiares, como la gran mayoría de estas personas reside en distintas provincias del país, la institución brinda un servicio de reservación de pasajes.

La reservación de pasajes no es más que un servicio que provee al usuario de un boletín en el cual está la información sobre la fecha, transporte y asiento que le han sido reservados para realizar su viaje; el encargado de gestionar estos pasajes es la Dirección de Transporte de la UCI en coordinación con las distintas Agencias de Viajes del país. De dos a tres veces en cada curso dicha Dirección tramita una reservación de pasajes para Pases Masivos de la institución, o sea, es un pase donde salen hacia sus provincias y municipios de residencia todos los estudiantes y la mayoría de los trabajadores y profesores.

En el momento en que un Pase Masivo comienza a ser tramitado se solicita la información sobre los viajeros que harán uso del servicio (Lista de Viajeros), con esta se obtiene la cantidad de viajeros y se gestionan los pasajes en la Agencia de Viajes correspondiente. Una vez que la Agencia reporta la información sobre los medios de transporte que ha puesto a disposición de la UCI, las fechas de salida y sus capacidades (ej. Tren: Coches y asientos por coche), se procede a la distribución, que no es más que ubicar a los viajeros por medio de transporte asignándoles un asiento según su destino. Esta distribución se hace siguiendo un criterio según se necesite, o sea, pueden distribuirse por tipo de persona (profesor, familiar, estudiante, trabajador, dirigente), por año, por facultad o por destino (municipio o provincia). También debe seguir cierto orden lógico, es decir en cada medio de transporte, dígase coche u ómnibus, se ubican los estudiantes y con ellos debe ir un profesor responsable y deben ubicarse a sus familiares con él en caso de tenerlos y ser posible, pero siempre dando prioridad a los estudiantes.

1.3 – Objeto de estudio

1.3.1 – Descripción general

La Dirección de Transporte de la UCI brinda el servicio de reservación de pasajes y es la encargada de la reservación de los Pases Masivos del centro. Dicho departamento se encarga además de corregir los errores que puedan haberse cometido e informar a los usuarios, las personas residentes en la UCI, sobre su ubicación en los medios de transporte. Por otra parte debe crear los siguientes informes sobre el Pase Masivo:

- Cantidades de Viajeros por provincia y municipios
- Listado con los datos de los Viajeros
- Listado de Distribución de los Viajeros con sus datos y la ubicación

Actualmente este servicio se brinda de forma manual, o sea no esta completamente automatizado por lo que son frecuentes los errores y el tiempo que toma corregirlos es demasiado en comparación con el tiempo disponible para esta tarea. Además los usuarios reciben la información de su ubicación y si hay errores debe hacerse una engorrosa búsqueda en listados impresos con la consiguiente perdida de tiempo y la dificultad para subsanar dichos errores.

1.3.2 – Descripción del negocio actual

La Dirección de Transporte de la UCI recibe la información de un nuevo Pase Masivo y comienza a gestionar con las cantidades de viajeros los transportes. Una vez logrado esto distribuye los pasajeros y genera los boletines. Si algún usuario desea saber su ubicación debe esperar a que sean entregados los boletines y es entonces que se detectan los errores. En caso de algún error los usuarios reclaman ya sea por correo electrónico, teléfono o acudiendo personalmente a la Dirección y deben esperar que el problema sea resuelto en un tiempo bastante largo.

1.3.3 – Situación problémica

La reservación de Pase Masivo no esta exenta de errores producto del manejo de mucha información por separado. El proceso de rectificación de errores es demasiado trabajoso debido a que se hace completamente manual y por tanto toma un tiempo considerable. Los usuarios no tienen una forma eficaz, rápida y cómoda que les permita conocer la información sobre la distribución del Pase Masivo y sus respectivas ubicaciones, fecha de salida y transporte.

1.4 - Sistemas automatizados existentes vinculados al campo de acción

1.4.1 – Sistema de Distribución

Este sistema se nutre de la información obtenida de la lista de personas a viajar entregada a la Dirección de Transporte y provee una Lista de Distribución que contiene los datos de los viajeros ya ubicados, asignándole a cada viajero un asiento en un medio de transporte y la hora y fecha de salida. Por otra parte permite imprimir estos datos en el formato de boletín, que es lo que será entregado a los usuarios para que puedan hacer uso de este servicio.

1.5 – Análisis comparativo de otras soluciones con la propuesta

El Sistema de Reservación existente presenta varias deficiencias como el hecho de que para modificar una distribución anteriormente hecha o recién finalizada es necesario hacerla nuevamente desde el principio, no permite hacer cambios en los datos de los usuarios, por ejemplo municipio y/o provincia, una vez que están en el sistema, no es posible obtener la información de un usuario especifico, o sea no es posible hacer búsquedas o emitir reportes. Sin embargo el sistema que se propone permite hacer todo tipo de reportes necesarios tanto para los usuarios como para los trabajadores además de que no necesita de una lista de personas a viajar pues él mismo es capaz de gestionar dicha lista y hasta hacer reportes con esta información. Brindando además un tratamiento seguro de los datos y una interfaz cómoda tanto para los trabajadores como para los usuarios.

1.6 – Objetivos generales y específicos

Este proyecto se propone automatizar completamente, de forma eficiente y a la altura de los requerimientos una solución a los problemas que presenta actualmente la Reservación de Pases Masivos de la UCI. Más específicamente la implementación de un Sistema de Reservación de Pases Masivos que automatice los procesos de recopilación de información y distribución de usuarios, que además sea capaz de distribuir los usuarios según el criterio dado por el trabajador encargado de la distribución, de dar un informe sobre la ubicación de los usuarios de forma general y por usuario específico y por último que guarde un historial de los Pases que se han realizado para que puedan ser consultados en caso de necesitarse esta información.

1.7 – Conclusiones

Este capítulo ha servido para explicar los problemas y características del objeto de estudio, los conceptos y definiciones que ayudan a comprender específicamente las condiciones del problema y a partir de esto obtener los objetivos, tanto generales como específicos, que permitirían darle una solución eficaz y elegante a la situación problémica.

Capítulo 2

Tendencias de las tecnologías actuales a tener en cuenta

2.1 – Introducción

En este capítulo se hace un análisis sobre las tendencias tecnológicas de las herramientas y lenguajes de programación utilizados en la implementación, así como el uso de RUP en el análisis y diseño del sistema. Con este análisis se pretende establecer la razón por la cual fueron escogidos para desarrollar el presente trabajo.

2.2 – Descripción de las tecnologías actuales en las que se basa la propuesta

2.2.1 - .NET

.NET es un proyecto de Microsoft que consiste en una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permite un rápido desarrollo de aplicaciones. Se podría considerar a .NET como una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, en competencia con la plataforma Java de Sun. Este proyecto es una estrategia horizontal de Microsoft que integra todos sus productos, desde el Sistema Operativo hasta las herramientas de Marketing, pasando por las de programación.

A largo plazo Microsoft pretende reemplazar la Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) Win32 o Windows API con la

plataforma .NET. Esto debido a que la API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows. La plataforma .NET pretende solventar la mayoría de estos problemas proveyendo un conjunto único y expandible con facilidad, de bloques interconectados, diseñados de forma uniforme y bien documentados, que permitan a los desarrolladores tener a mano todo lo que se necesita para producir aplicaciones sólidas.

Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .Net, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y librerías de clases adicionales); siendo algunas de ellas iniciativas de distribución gratuita bajo la licencia GNU.

2.2.2 - .NET Framework, arquitectura básica de la plataforma .Net.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución distribuido.

Los principales componentes de este entorno son los lenguajes de compilación, la Biblioteca de clases de .Net y el Entorno Común de Ejecución de Lenguajes o Common Language Runtime (CLR).

2.2.3 - Entorno Común de Ejecución de Lenguajes

2.2.3.1 – Common Language Runtime (CLR)

El CLR es el verdadero núcleo de la plataforma .Net, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo (W2k y W2003).
La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un código intermedio (MSIL, Microsoft Intermediate Language). Para generar dicho código el compilador se basa en Common Language Specification (CLS) que determina las reglas necesarias para crear ese código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso, un compilador JIT (Just-In-Time) es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .Net independencia de la plataforma hardware, que no de sistema operativo. La compilación JIT la realiza el CLR a medida que el programa invoca métodos, el código ejecutable obtenido, se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

La funcionalidad del sistema Microsoft .NET se basa en las capacidades de la Infraestructura del Lenguaje Común (CLI, Common Language Infraestructure), a diferencia de algunos sistemas basados en máquinas virtuales. El CLI fue diseñado desde el principio para soportar una variedad de lenguajes de programación. Se espera también que el CLI este disponible en varias plataformas y no solo Windows. Esta combinación de capacidades

con múltiples lenguajes y una implementación multiplataforma hacen que el CLI sea un objetivo importante para los compiladores futuros. Las ideas de máquinas virtuales, lenguajes intermedios y lenguajes con ejecución independientes de la plataforma es una idea muy popular entre los investigadores desde hace mucho tiempo.

2.2.3.2 – CLR y COM

A diferencia de COM, el CLR se crea desde el principio con un formato completamente especificado para describir, los contratos de los componentes. Este formato es llamado en forma genérica "metadata". El CLR provee facilidades para leer y escribir la metadata programáticamente sin el conocimiento del formato de archivo usado, esto quiere decir que el programador puede expresar los atributos desde el código, es una programación basada en atributos, por ejemplo los atributos de transacción de un componente puede ser expresados explícitamente y el CLR puede verificar de que se provea el ambiente para su ejecución. CLR metadata es claro y extensible vía atributos personalizados, los cuales a su vez están basados en tipos. CLR metadata contiene la información sobre la dependencia de los componentes y la versión, lo cual presenta nuevas técnicas para manejar las versiones en los componentes. Finalmente la presencia de la metadata es obligatoria, lo cual hace que la infraestructura y las herramientas basadas en el CLR sean más fáciles de usar y de construir que en ambientes donde la metadata es opcional, finalmente aquí hay un solo formato y una sola forma de representar los contratos.[18]

La segunda forma en que los contratos en el CLR difieren de los contratos en COM es en la naturaleza misma de los contratos. En COM, un contrato de un componente implica una representación precisa en memoria de una tabla virtual por ejemplo, y cualquier otra estructura de datos que son pasadas como métodos de los parámetros, los cuales ya tienen una expresión fija en

términos de direcciones de memoria. Los contratos en el CLR describen la estructura lógica de los tipos. Los contratos en el CLR específicamente no describen la representación en memoria. El CLR pospone las decisiones acerca de estas consideraciones de representación en memoria hasta que el tipo (o componente) se carguen por primera vez en tiempo de ejecución (Just in Time).

Otra facilidad del CLR es que soporta la programación generativa que es llamada CodeDOM. El CodeDOM permite que los programas en C#/VB .NET/JavaScript sean construidos en memoria como un modelo de objetos fuertemente basado en tipos. El CodeDOM permite a los programas generativos que emiten código fuente posponer la decisión sobre el lenguaje de salida hasta el último momento.[18]

2.2.4 - Biblioteca de Clases Base de .Net

La Biblioteca de Clases Base (BCL por sus siglas en inglés) maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones, incluyendo entre otras:

- Interacción con los dispositivos periféricos
- Manejo de datos (ADO.NET)
- Administración de memoria
- Transmisión y recepción de datos por distintos medios (XML, TCP/IP)
- Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET)
- Manejo y administración de excepciones
- Manejo del sistema de ventanas
- Herramientas de seguridad e integración con la seguridad del sistema operativo
- Manejo de tipos de datos unificados

- Interacción con otras aplicaciones
- Manejo de arreglos de datos y colecciones
- Manipulación de archivos de imágenes
- Generación de código
- Auto descripción de código
- Interacción con el API Win32 o Windows API
- Compilación de código

Esta funcionalidad se encuentra organizada por medio de espacios de nombres jerárquicos.
La Biblioteca de Clases Base se clasifica, en tres grupos clave:

- ASP.NET y Servicios Web XML
- Formularios Windows
- ADO.NET

2.2.5 – ADO.NET

2.2.5.1 – El nuevo ADO

ADO.NET siempre fue uno de los componentes más notables dentro de la nueva generación de aplicaciones .NET. Es decir, es bastante sencillo entender la evolución natural de una herramienta de desarrollo de una versión a otra, por ejemplo una nueva versión de Visual Basic, sin embargo las implicaciones que tiene un nuevo "motor" de acceso a datos son enormes, sobre todo considerando que el 80% de las aplicaciones escritas acceden a alguna fuente de datos.[6]

Antes que nada, ADO.NET ahora es parte del .NET Framework, esto quiere decir que es, de alguna manera, parte del Sistema Operativo y no más un redistribuible de 4 o 5 MB que se necesita alojar en la PC del cliente o junto al instalador de una aplicación. Este hecho representa una enorme ventaja

debido a que los desarrolladores pueden enfocarse en utilizar el acceso a datos y no tanto en cómo se desplegará a los clientes la librería.[6]

Una ventaja que ADO.NET brinda al ser parte del .NET Framework es que todos los lenguajes compatibles con .NET utilizan las mismas clases para acceder y manipular datos, únicamente es la sintaxis la que difiere en cada lenguaje de desarrollo que se utiliza. Por ejemplo, VB, C# o cualquier lenguaje compatible con NET, abrirá una conexión a una base de datos MsSQL Server 2000 utilizando exactamente la misma librería, *System.SqlClient.SqlConnection* en este caso.

2.2.5.2 - Modelo ADO.NET

ADO.NET es bastante más amigable con MS SQL Server que antes, basta solo con utilizar la librería *System.Data.SqlClient.SqlConnection*, escrita específicamente para MSSQL Server 7.x o posterior, para obtener mayores ventajas que simplemente omitiendo capas de conversión. ADO.NET es capaz de utilizar internamente TDS (Tabular Data Stream), el mismo mecanismo de comunicación que SQL Server 7.x y posterior utiliza internamente, ahorrando tiempo y recursos evitando realizar conversiones de ADO a OLEDB y de OLEDB a ODBC y de ODBC a la fuente de datos. [6]

Sin embargo, para los que no tengan la suerte de contar con un servidor de base de datos MSSQL Server 7.0 o posterior, ADO.NET utiliza su clase System.Data.Oledb. Gracias a esta biblioteca, Microsoft soporta acceso a diferentes fuentes bajo el mismo modelo de acceso a datos. Aunque se mantienen separados los objetos SqlClient y OleDb, ambos implementan clases y objetos similares entre ellos lo que hace muy sencillo trabajar con los mismos.

2.2.5.3 - Conectado o Desconectado

Microsoft ha puesto en su lista de prioridades para ADO.NET el que cada vez los desarrolladores puedan escribir aplicaciones más y más escalables, y para hacerlo se debe reducir al máximo el número de "conexiones abiertas" en la base de datos. En ADO.NET, gracias a los DataSets, es posible tomar una parte de los datos y mantenerlos en memoria en forma relacional, tal cual son almacenados, esto permite brindarle datos a la aplicación cliente sin sacrificar el rendimiento al mantener conexiones abiertas a la base de datos.

Contrariamente a lo que tal vez se pueda pensar, no todo es "desconectado" dentro del mundo de ADO.NET. Microsoft sabe que un gran volumen de aplicaciones por sus requerimientos y por la naturaleza del negocio requieren una arquitectura conectada, mas aún, en aplicaciones que cuentan con un alto volumen de actualizaciones de datos y necesitan contar con los datos frescos al instante, los DataSets no son la mejor opción, siendo el modelo "conectado" el que mejor se adapta a este escenario.

El objeto bandera del modelo conectado en ADO.NET se llama DataReader, cada clase implementa uno propio, SQLDataReader u OledbDataReader, ambos básicamente hacen lo mismo: proporcionar un objeto simple, rápido y liviano donde poder recibir los datos que envía la Base de Datos y procesarlos, un registro a la vez.

Entonces se tienen dos escenarios, ambos tienen ventajas y desventajas. Si bien el escenario "conectado" simplifica muchas tareas como el extraer directamente los datos y manipularlos rápidamente con el objeto DataReader, también limita la escalabilidad porque es mayor el numero de veces que se realizaran consultas a la base de datos, siendo necesario incluso el recurrir a comandos adicionales para enviar los cambios realizados a la información, ya sea vía Procedimientos Almacenados o sentencias SQL. En el lado contrario, el modelo conectado trabaja tranquilamente con una

porción desconectada de los datos, gracias a esto, se puede crear vistas, navegar dentro del DataSet, vincularlo a cualquier control y otras funcionalidades, el problema radica generalmente al momento de vaciar los datos a la fuente de datos, debido a que estos pudieron ser alterados, modificados o incluso borrados por otros usuarios mientras se trabaja en forma desconectada. Los DataSets también involucran un trabajo más minucioso de diseño de base de datos, debido a justamente a la naturaleza desconectada con los que los datos serán extraídos y actualizados posteriormente.

2.2.5.4 - Aplicaciones Distribuidas

Microsoft ha visto evolucionar en los últimos años las aplicaciones distribuidas. Cada vez son más los negocios que necesitan desplazar información entre componentes dentro y fuera de la empresa. ADO.NET ofrece una ventaja tremenda respecto a su antecesor, ya que es capaz de representar repositorios como DataSet en forma de XML.

ADO.NET a través de los Datasets provee repositorios inteligentes de datos que pueden comportarse mucho mejor que los tradicionales Recordsets o arreglos. No producen sobrecargas debido a que lo que transmiten es texto XML y no representaciones de objetos los cuales son más complejos y pesados.

2.2.6– ASP.NET

ASP.NET es un ambiente de programación construido sobre el entorno NGWS (Servicios de la Nueva Generación de Windows, siglas en inglés), que permite crear poderosas aplicaciones de Internet e Intranet. Ofrece además varias ventajas importantes sobre los modelos previos de desarrollo para Internet

2.2.6.1 – Más Eficiencia

ASP.NET corre código compilado sobre el entorno NGWS en el servidor. Distinto a sus predecesores interpretados, ASP.NET usa amarres tempranos ("early binding"), así como compilación justo a tiempo ("just-in-time compilation"), optimización nativa, y servicios de caché, sin configuración adicional. Para los desarrolladores, esto significa eficiencia dramáticamente superior antes de escribir la primera línea de código. [7]

2.2.6.2 - Poder y Flexibilidad

Dado que ASP.NET está basado en el Entorno Común de Ejecución de Lenguajes o Common Language Runtime (CLR), el poder y la flexibilidad de la plataforma completa está disponible para los desarrolladores. Las librerías de Clases del CLR, la Mensajería, y las soluciones de Acceso a Datos, son accesibles al través de Internet. ASP.NET permite el uso de una gran variedad de lenguajes de programación y, por tanto, garantiza que no haya pérdidas en el desarrollo de aplicaciones COM cuando se migra hacia ASP.NET.

2.2.6.3 – Simplicidad

ASP.NET hace fácil el ejecutar tareas comunes, desde el simple envío de un formulario o la autenticación de un cliente, hasta el despliegue y la configuración de un sitio Web. Por ejemplo, el entorno de paginado de ASP.NET permite construir interfases de usuario que separan limpiamente la lógica de la aplicación del código de la presentación, y maneja eventos con un modelo sencillo de procesamiento de formularios al estilo de Visual Basic. Adicionalmente, el CLR simplifica el desarrollo con servicios de código gerenciado, como el conteo automático de referencias y la limpieza automática de la memoria utilizada por la aplicación.

2.2.6.4 – Gerenciabilidad

ASP.NET usa un sistema jerárquico de configuración, basado en archivos de texto, que simplifica la aplicación de parámetros de configuración al servidor sus aplicaciones. Como la información de configuración es almacenada en forma de texto, nuevos parámetros pueden ser configurados sin recurrir a herramientas de administración locales. Esta filosofía de "cero administración local" también se extiende al despliegue de aplicaciones de ASP.NET. Una aplicación de ASP.NET se despliega a un servidor simplemente copiando los archivos necesarios al servidor. No hay que reiniciar el servidor, ni siquiera para reemplazar código compilado que ya está en servicio.

2.2.6.5 - Escalabilidad y Disponibilidad

ASP.NET ha sido diseñado para la escalabilidad con características específicamente dirigidas a mejorar el funcionamiento de servidores racimados (clustered) y de servidores con procesadores múltiples. Los procesos del servidor son vigilados y administrados por el entorno del ambiente de ejecución de ASP.NET, así que si algún proceso se entorpece o se detiene, un nuevo proceso puede ser creado para reemplazarlo, lo cual ayuda a mantener la disponibilidad de la aplicación para manejar solicitudes de servicio.

2.2.7 – Servicios Web XML

A pesar de su sencillez aparente, XML está transformando completamente la creación y el uso de software. La Web revolucionó la comunicación entre usuarios y aplicaciones. XML está revolucionando la comunicación entre aplicaciones o, de forma más general, la comunicación entre equipos, pues ofrece un formato de datos universal que permite adaptar o transformar fácilmente la información, los servicios Web XML utilizan protocolos estándar,

permiten que las aplicaciones compartan distintos tipos de información, son unidades de código discretas, cada una de las cuales se encarga de un conjunto limitado de tareas, están basados en XML, el lenguaje universal del intercambio de información en Internet y pueden utilizarse en cualquier plataforma o sistema operativo, independientemente del lenguaje de programación utilizado. Sus beneficios radican en su arquitectura descentralizada, su profunda estandarización que los vuelve independientes de la plataforma, su disponibilidad abierta al público y la facilidad y uniformidad de acceso.

Los servicios Web XML también permiten que los programadores puedan elegir entre generar todas las partes de sus aplicaciones o utilizar servicios Web XML creados por otros. De este modo, una empresa no necesita crear todas las partes de una solución completa. Pero no solo permiten que las aplicaciones compartan información sino que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado las aplicaciones, no importa el sistema operativo o la plataforma en que se ejecutan y cuáles los dispositivos utilizados para obtener acceso a ellas. Aunque los servicios Web XML son independientes entre sí, pueden vincularse y formar un grupo de colaboración para realizar una tarea determinada.

Estos servicios pueden ser invocados por medio de protocolos estándar tales como SOAP, XML y UDDI. Estos protocolos son definidos por organizaciones de estándares públicos como el consorcio W3C.

2.3 – Fundamentación del uso del Proceso Unificado de Desarrollo

2.3.1 – Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo o Rational Unified Process (RUP) es un proceso de ingeniería de software que enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos los miembros del equipo.

Para desarrollar un sistema es necesario coordinar el trabajo en un proceso que integre las diferentes fases que componen el desarrollo de la aplicación. Es necesario un método común que provea una guía para ordenar las actividades del equipo de trabajo, dirija las tareas de todo el equipo y de cada miembro por separado, especifique qué debe ser desarrollado en cada momento y brinde criterios para la medición y el control de los productos y actividades. Todo esto se logra utilizando RUP, un proceso de desarrollo de aplicaciones, o sea es el conjunto de actividades que partiendo de los requisitos de un usuario es capaz de concebir un sistema. El Proceso Unificado está basado en componentes, por lo que el sistema en desarrollo estará formado por componentes muy interrelacionados entre sí mediante interfaces estrictamente definidas. [13]

El RUP tiene un desarrollo de software dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental que pretende implementar las mejores prácticas en ingeniería de software, con el objetivo de asegurar la producción de software de calidad, dentro de plazos y presupuestos predecibles:

- Desarrollo iterativo de software
- Administración de requerimientos
- Uso de arquitecturas basadas en componentes
- Modelación visual del software
- Verificación de la calidad del software
- Control de cambios

2.3.1.1 - Desarrollo iterativo

Hoy en día no es realista utilizar un modelo lineal de desarrollo, como por ejemplo el método de cascada, ya que el software moderno es complejo y novedoso. Un proceso iterativo, al mismo tiempo que el sistema va desarrollándose, permite una comprensión creciente de los requerimientos. Con RUP se logran reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente siguiendo un modelo iterativo o de ciclos que aborda las tareas más riesgosas primero.

2.3.1.2 - Administración de requerimientos

Utilizando RUP se pueden obtener, organizar y documentar los requerimientos de funcionalidad y restricciones, rastrear y documentar las decisiones, además de captar y comunicar los requerimientos del negocio. Utilizando los casos de uso y escenarios indicados por el proceso, se logra una forma óptima de captar requerimientos y guiar tanto el diseño como la implementación y las pruebas.

2.3.1.3 - Arquitecturas basadas en componentes

El proceso tiene como fundamento el diseño en una época temprana en el tiempo de vida del proyecto una arquitectura base ejecutable que sea flexible, fácil de modificar, comprensible y debe promover la reutilización de componentes. RUP apoya el desarrollo basado en componentes nuevos y ya existentes.

2.3.1.4 - Verificación de la calidad del software en RUP

La funcionalidad de un sistema a desarrollar es esencialmente importante pero no se deben descuidar el rendimiento y la confiabilidad. En función de esto RUP brinda facilidades a la hora de planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifiquen estas cualidades. El aseguramiento de la calidad forma parte del proceso de desarrollo y no es una responsabilidad de un grupo independiente como erróneamente se piensa algunas veces.

2.3.1.5 - Control de cambios

Cuando se esta desarrollando una aplicación son inevitables los cambios, pero es necesario evaluar si éstos son necesarios y también es necesario conocer lo que dichos cambios implicarán. RUP indica como controlar, rastrear y supervisar los cambios que puedan ocurrir dentro del proceso iterativo de desarrollo del proyecto.

2.3.1.6 - Modelado visual

El Proceso Unificado de Desarrollo propone un modelamiento visual de la estructura y el comportamiento de la arquitectura y los componentes. Creando un esquema donde los bloques de construcción deben ocultar detalles, permitir la comunicación entre los integrantes del equipo de desarrollo y permitir analizar la consistencia entre los componentes, el diseño y la implementación. La base del modelamiento visual de RUP esta dada por el Lenguaje de Modelación Unificado, más conocido por su nombre en ingles Unified Modeling Language (UML). Este es el encargado de preparar todos esquemas de un sistema en desarrollo y es una parte fundamental del Proceso Unificado de Modelado.

2.3.2 – Lenguaje Unificado de Modelado

La clásica cuestión de por qué realizar un análisis y diseño de la aplicación si lo que el usuario necesita es que funcione y satisfaga sus expectativas, marca la línea entre lo que el desarrollador debe hacer y lo que el usuario final y quienes encargan el producto, piensan que debe hacerse. La búsqueda de fórmulas para mejorar la concepción y realización de una aplicación por quienes conforman el equipo de desarrollo, encuentra en este lenguaje de modelado una herramienta valiosa.

El Lenguaje Unificado de Modelado, en adelante UML (Unified Modeling Language), es el resultado más integrador de una serie de métodos de análisis y diseño orientado a objetos. Se origino entre fines de los ochenta y principios de los noventa, UML no fue concebido como un método en sí mismo, sino como la notación básicamente gráfica de la que se puede valer cualquier método para expresar los diseños y el proceso que orienta los pasos a dar para realizar este diseño. Así completa lo que todo método debe presentar, un lenguaje de modelado y un proceso (RUP). [14]

Al igual que ocurre con el mundo real que cada vez se vuelve más complejo, los sistemas asistidos por computadora también aumentan cada día su complejidad. A menudo tienen implicados múltiples partes de hardware y software, conectados en red a través de grandes distancias, vinculadas a bases de datos que contienen enormes cantidades de información. La clave para organizar todo el proceso es diseñar una forma sobre la cual el cliente, los analistas y los desarrolladores puedan entenderse y ponerse de acuerdo en el desarrollo del sistema. El UML proporciona está organización.

Utilizando UML la aplicación es diseñada y documentada antes de ser codificada, sabiendo así lo que se logrará por adelantado, también permite descubrir con facilidad el código reutilizable y hacer un mejor uso de este pues el sistema existe antes de crear la primera parte del código; cualquier

descuido o error en el diseño del sistema puede ser descubierto a tiempo en los diagramas de diseño y así el software se comportará de la manera que se espera y no habrá sorpresas, comoquiera que el diseño total del sistema dicta el modo en que se desarrollará la aplicación las decisiones finales serán tomadas antes de que aparezca algún código mal escrito y con esto ahorrar tiempo en el desarrollo. Por otra parte al hacer modificaciones en el sistema, es mucho más fácil hacerlo sobre la documentación UML y si se incorporan nuevos integrantes al equipo de desarrollo, los diagramas UML les permitirán rápidamente ponerse al corriente de la idea del proyecto, además de hacer más eficiente la comunicación con los desarrolladores tanto externos como internos.

El proceso de desarrollo de un sistema informático implica varias personas, el cliente, que es quien tiene el problema que ha de ser solucionado, el analista, que documenta el problema del cliente y lo trasmite a los desarrolladores, los programadores implementan la aplicación, la prueban y la instalan sobre el hardware del cliente. Hoy en día los sistemas se han hecho tan complejos que una sola persona no puede conocer todas las facetas de las necesidades de un negocio, entender estas necesidades, diseñar una solución para ellas, escribir el programa e instalarlo asegurándose de que todos los componentes trabajen correctamente.

2.4 – Fundamentación del lenguaje y herramientas utilizadas

2.4.1 – Visual Studio .NET

Visual Studio .NET es una herramienta de nueva generación que aumenta la productividad del programador ya que contiene todo lo necesario para generar e integrar rápidamente aplicaciones y servicios Web XML. Esta herramienta junto al .NET Framework permiten al desarrollador hacer servicios Web basados en XML además de permitir desarrollar proyectos en variados lenguajes pues es capaz de compilar códigos programados en C#

(C Sharp), Visual Basic, C++, J#, JScript y algunos otros que han sido adaptados como Perl, Phyton y Cobol. Con esto hace posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes.[9]

La Plataforma .NET viene incorporada directamente en la nueva línea de sistemas operativos Windows .NET. También se ha creado la Plataforma Compacta de .NET (.NET Compact Framework) para dispositivos móviles.

Esta plataforma ha sido diseñada con una arquitectura abierta que permite a los programadores aprovechar los conocimientos de programación actuales ya que pueden utilizar cualquier lenguaje orientado a Microsoft .NET Framework y así evitan los costosos cursos de reciclaje sin mencionar el tiempo que les ahorra. Al estar Visual Studio .NET incorporado al Framework, la más reciente plataforma de servidor de Microsoft Windows, provee gran escalabilidad, confiabilidad y seguridad a las aplicaciones. También se han simplificado la administración y la implementación de las aplicaciones en un ambiente de producción, reduciendo así los costos totales del ciclo de desarrollo.

2.4.2 – Lenguaje C Sharp (C#)

Los primeros rumores de que Microsoft estaba desarrollando un nuevo lenguaje de programación surgieron en 1998, se decía de un lenguaje que entonces llamaban COOL y que era muy similar a Java. En junio de 2000, Microsoft despejó todas las dudas liberando la especificación de un nuevo lenguaje llamado C#. A esto le siguió rápidamente la primera versión de prueba del entorno de desarrollo estándar (SDK) .Net, que incluía un compilador de C#. El nuevo lenguaje estaba diseñado por Anders Hejlsberg (creador de Turbo Pascal y arquitecto de Delphi), Scott Wiltamuth y Peter

Golde. Estos describieron el lenguaje como simple, moderno, orientado a objetos, de tipado seguro y con una fuerte herencia de C/C++. [15]

Éste nuevo lenguaje orientado a objetos con énfasis en Internet se basa en las lecciones aprendidas de los lenguajes C, C++, Java y Visual Basic. Por lo tanto se trata de un lenguaje que combina todas las cualidades que se pueden esperar de un lenguaje moderno (orientación a objetos, gestión automática de memoria, etc.) a la vez que proporciona un gran rendimiento.

El lenguaje es sencillo, tomando el mismo patrón de los lenguajes de programación modernos, incluye un amplio soporte de estructuras, componentes, programación orientada a objetos, manipulación de errores y recolección de basura entre otros, que es construido sobre los principios de C++ y Java. C# contiene las herramientas para definir nuevas clases, que son la base de los lenguajes de programación orientados a objetos, sus métodos y propiedades, al igual que la sencilla habilidad para implementar encapsulación, herencias y polimorfismo, los cuales son los tres pilares de la programación orientada a objetos.

Este lenguaje tiene un nuevo estilo de documentación XML que se incorpora a lo largo de la aplicación lo cual simplifica la documentación en línea de clases y métodos, soporta también interfaces, una forma de estipular los servicios requeridos de una clase. Las clases en C# pueden heredar de un padre pero puede implementar varias interfaces. Provee también soporte para estructuras. Una estructura es un tipo restringido que no exige tanto del sistema operativo como una clase. Una estructura no puede heredar ni dar herencias de clases pero puede implementar una interfaz. Brinda además características de componentes orientados, como propiedades, eventos y construcciones declaradas (atributos). La programación orientada a componentes es soportada por el CLR. Aunque no esta diseñado para trabajar con punteros del mismo modo que C++ esto no quiere decir que no puedan utilizarse ya que C# permite acceder directamente a la memoria

usando el estilo de punteros de C++ en caso de ser especificado como "código no seguro". Este código queda marcado y no se ejecuta en ámbitos donde no tenga permisos.

C# combina los mejores elementos de múltiples lenguajes de amplio uso como C++, Java, Visual Basic o Delphi. De hecho su creador, Anders Heljsberg, fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás de este lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los programadores de C/C++/Java sea lo más inmediata posible.

El .NET Framework unido al C# permite el desarrollo de aplicaciones Web de forma rápida, brindando seguridad y eficiencia ya que Visual Studio .NET proporciona una amplia gama de controles y útiles para la interfaz Web especialmente dirigidos al desarrollo de aplicaciones para Internet e Intranet.

2.4.3 – Embarcadero ERStudio

Embarcadero ERStudio es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos y desarrolladores construyen y mantienen aplicaciones complejas de bases de datos grandes. Está equipado además para crear y manejar diseños de bases de datos funcionales y confiables y ofrecer fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación basada en HTML y fácil creación de reportes, reingeniería inversa de bases de datos y un repositorio para el modelado.

Esta herramienta se utiliza para el diseño y la construcción tanto lógica como física de base de datos. Está diseñada para hacer más fácil de entender el estado actual de los datos del proyecto. Simple y fácil de usar, ayuda a tomar decisiones en cómo resolver embotellamientos de los datos y elimina redundancia.

2.4.3.1 - Potencial de ERStudio

Si se está comenzando un nuevo diseño o está manteniendo una base de datos existente, ERStudio se combina con las características para ayudar a conseguir un trabajo eficiente. La creación de diagramas es clara y rápida. Tiene la posibilidad de realizar diagramas con desempeño rápido. También es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Es posible crear subvistas para separar y manejar áreas importantes. ERStudio automáticamente mantiene todas las dependencias entre subvistas y el diagrama completo. Ya sea que se inicie un nuevo diseño o se mantenga uno existente, ERStudio está equipado con elementos de ayuda para hacer el trabajo de manera efectiva.

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se desnormalizan los diseños físicos para su desarrollo. Por otra parte mantiene ligaduras entre todos los niveles de su diseño por lo tanto puede hacer cambios en cualquier dirección entre ellos. Permite realizar un chequeo de la normalización y la compilación con la sintaxis de la plataforma de la base de datos.

Es capaz de generar objetos de base de datos como vistas, procedimientos almacenados, reglas y tipos de datos, lo cual ayuda en el ordenamiento de

tipos de objetos para eliminar errores de dependencia al construir la base de datos.

Una vez que se ha diseñado la base de datos, se pude construir o generar código fuente para todo o para parte de los diseños de la base de datos. Propiamente hace la secuencia de la creación de tipos de objetos diferentes para asegurar eficiencia, y construir bases de datos libres de errores. Actualiza una base de datos a partir del diagrama, permite aplicar cambios de diseño del modelo de datos directamente a la base de datos. Cuando se comparan las diferencias entre los dos, formula una estrategia de alteración inteligente que implementa el diseño de las modificaciones mientras se preserva la tabla con los datos existentes, privilegios de objetos, y dependencias en la base de datos.

Cuenta con ingeniería de reverso, cuando se necesite iniciar un trabajo de una base de datos existente, ERStudio puede hacer una ingeniería de reverso al esquema completo para cualquier plataforma de bases de datos. La operación de la ingeniería de reverso extrae eficientemente definiciones de objetos y construye un modelo de datos gráfico.

2.4.4 – Microsoft SQL (MsSQL) Server 2000

Microsoft SQL Server 2000 brinda varias propiedades que permiten crear y administrar aplicaciones de base de datos, permitiendo establecer propiedades para secuencias de comandos, configuraciones regionales, tipos de datos, instrucciones Transact-SQL, intercalaciones y otras características.

Una base de datos de MsSQL Server 2000 proporciona una separación lógica de los datos, aplicaciones y mecanismos de seguridad. Una instancia de SQL Server admite varias bases de datos. En un único equipo, puede haber varias

instancias de SQL Server. Cada instancia de SQL Server puede tener varias bases de datos.

Cada base de datos admite grupos de archivos, que ofrecen la posibilidad de distribuir de forma física la ubicación de los datos. Un grupo de archivos de SQL Server clasifica los ficheros del sistema operativo que contienen datos desde una única base de datos de SQL Server para simplificar tareas de administración de las bases de datos, como las copias de seguridad. Un grupo de archivos es propiedad de una base de datos de SQL Server y no puede contener archivos de sistema operativo de más de una base de datos, aunque una única base de datos puede contener más de un grupo de archivos. Una vez creada una base de datos, es posible agregar a ella grupos de archivos.

MsSQL Server instala también de forma predeterminada las siguientes bases de datos:

- La base de datos **model** es la plantilla utilizada en todas las bases de datos creadas por el usuario.
- La base de datos **tempdb** es similar a las tablas temporales de Oracle utilizadas para las tareas de almacenamiento y ordenación temporales, con la única diferencia que en SQL Server las tablas temporales se eliminan automáticamente al finalizar la sesión.
- La base de datos **msdb** es compatible con el Agente SQL Server y sus trabajos programados, alertas e información de réplica.
- Las bases de datos **pubs** y **Northwind** se utilizan como ejemplos para la formación.

2.4.4.1 – Microsoft SQL Enterprise Manager

MSSQL Enterprise Manager es la herramienta que permite administrar el Microsoft SQL Server 2000, con la cual es posible, entre otras cosas, definir grupos de servidores que

estén prestando servicio de SQL Server, registrar servidores individuales en un grupo, configurar todas las opciones para cada servidor registrado, crear y administrar todas las base de datos, usuarios, permisos y objetos para cada servidor registrado, definir y ejecutar todas las tareas administrativas en cada servidor registrado, diseñar y probar consultas SQL, procedimientos almacenados y scripts interactivamente invocando al Analizador de Consultas de SQL (SQL Query Analyzer) e invocar los asistentes definidos para MsSQL Server.

2.4.5 – Adobe Photoshop

Adobe Photoshop no es solo una simple herramienta fotográfica sino una poderosa máquina de retoques y perfeccionamiento por excelencia, que presenta posibilidades de descarga de pluggins (actualizaciones y componentes) de la red y una gran compatibilidad a la hora de hacer páginas Web. Permite el trabajo de montajes con la posibilidad de añadir capas y efectos que, conjugados con la manipulación de las tres dimensiones, produce un resultado tan interesante como refinado en el trabajo con imágenes. En resumen es una herramienta sumamente poderosa en el mundo del diseño.

2.4.6 – Rational Rose

Es una herramienta que permite la administración de actividades, recursos y resultados que forman parte de la disciplina de control de calidad de software. Esta herramienta brinda la posibilidad de organizar, asignar y programar la ejecución de las pruebas (manuales o automatizadas), además provee al líder del proyecto la posibilidad de evaluar el nivel de calidad del producto bajo pruebas con estadísticas de cobertura.

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa las especificaciones del UML.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.

Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Rational Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo. También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML y realizar Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

2.5 – Conclusiones

A lo largo de este capítulo se ha hecho un breve estudio y descripción de las tendencias de las tecnologías actuales así como una evaluación del RUP, además del lenguaje y herramientas utilizados en la implementación y diseño

de este proyecto, concluyendo que, para que el mismo pudiese dar respuesta al problema que se plantea, se desarrollaría una aplicación Web ya que su principal función es brindar un servicio y que estaría implementado sobre .NET Framework dado que es una plataforma con muchas prestaciones para los sistemas Web, además se utilizaría la tecnología ASP.NET, específicamente el lenguaje C# y para el trabajo con datos el MsSQL Server 2000 ya que es uno de los mejores y más conocidos gestores de bases de datos del mundo de la informática. Por otra parte que se utilizara Adobe Photoshop para el diseño de la interfaz Web

.

Capitulo 3

Descripción de la solución propuesta

3.1 – Introducción

Este capítulo ha sido dedicado por completo a la descripción de la modelación del problema al que se le propone solución. Se describen los procesos del negocio, actores, modelo de objetos, diagrama de clases, requisitos, la descripción de la propuesta de sistema, en fin se detalla todo el modelado de la solución propuesta.

Antes de continuar se ha considerado esclarecer algunas definiciones de suma importancia y que son necesarias para la correcta comprensión del tema:

Pase Masivo: Sucede cuando la gran mayoría de las personas que viven en la Universidad de las Ciencias Informáticas (UCI) salen hacia sus provincias de residencia, generalmente ocurre dos veces al año.
Familiar: Son los familiares de algún profesor, dirigente o trabajador que radiquen en la UCI.
CUJAE: Ciudad Universitaria José Antonio Echeverría.
Otros Viajeros: Es personal que no radica en la UCI pero por diversos motivos viajarán en los transportes destinados al Pase Masivo. Ej. Estudiantes de la CUJAE.
Viajeros: Son las personas que viajarán en un Pase Masivo determinado, pueden ser estudiantes, dirigentes, profesores, familiares, trabajadores u otros viajeros.

Boletín: Documento oficial que se le otorga a cada viajero cuando ocurre un Pase Masivo el cual contiene sus datos personales y los datos del medio de transporte donde viajará (asiento, hora y fecha de salida, tipo de transporte).
Reservación: Es la acción que tiene como resultado un boletín para el viajero.
Dirección de Transporte: Es el departamento de la UCI que atiende todo lo referente a transportación de personal, en este caso el Pase Masivo.
Transporte: Se refiere al Tren o Flota de Ómnibus según sea el caso.
Medio de Transporte: Son los que componen un Transporte, pueden ser coches u ómnibus y además tienen un destino (municipio o provincia) y asientos.
Distribución: Es el proceso de ubicación de los viajeros en los distintos medios de transporte según su destino y los criterios dados.
(A/B/C): Significa que se requiere A o B o C
(A, B, C): Significa que se requiere A, B y C
Cliente: Trabajador que utiliza el sistema
Usuario: Cualquier persona que utilice el sistema para obtener cierta información

3.2 – Reglas del negocio a considerar

En el negocio de Reservación de Pases Masivos debe tenerse en cuenta que siempre que se comienza una Distribución es vital ubicar primeramente a los estudiantes tratando de ubicar al menos un profesor por medio de transporte. Debe tratarse de que los profesores y sus familiares sean ubicados contiguamente siempre que sea posible y finalmente ubicar a los demás viajeros. En todo caso la prioridad para ser ubicados la tienen los estudiantes. Por cada viajero debe generarse un boletín como punto final de la distribución.

3.3 – Procesos del Negocio Actual

Llegado el momento de realizar el Pase Masivo comienza la Distribución del Pase Masivo de la UCI donde se pueden diferenciar dos procesos principales la Gestión del Pase Masivo, en este proceso es donde se reservan los pasajes, y la Distribución, que es donde se ubican los viajeros por medio de transporte. Esto se puede observar en el siguiente diagrama:

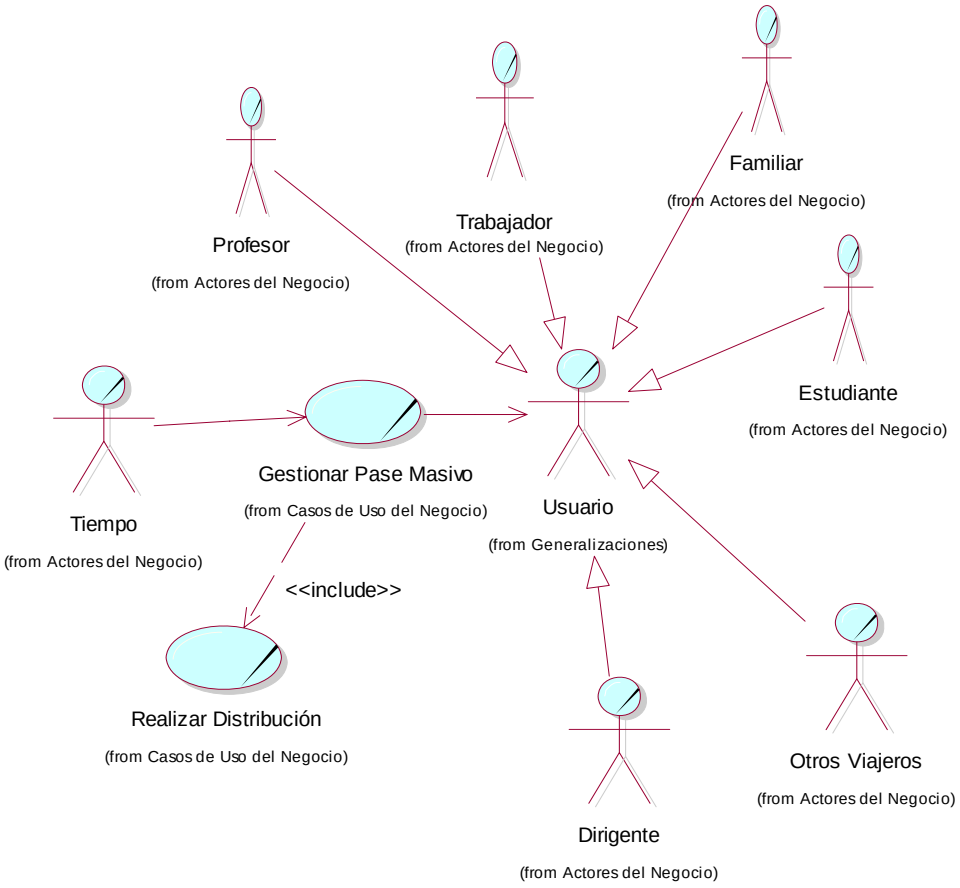


Figura 1. Diagrama de casos de uso del negocio

Descripción de actores y trabajadores del negocio:

Actores del Negocio	Descripción
Tiempo	Al terminar una etapa del curso docente se procede a la Gestión de un Pase Masivo.
Usuarios (Estudiantes, Profesores, Familiares, Dirigentes, Otros Viajeros, Trabajadores)	Son todas aquellas personas residentes en la UCI o no que se benefician del negocio, ya sea para viajar o simplemente para obtener información o reportes del mismo.

Trabajadores del Negocio	Descripción
Agencia	Es la Agencia de Viajes a la cual se le compran los pasajes necesarios para los viajeros.
Director de Transporte	Es el encargado de comenzar a Gestionar el Pase Masivo, obteniendo las cantidades de viajeros y solicitando a la Agencia los transportes para luego realizar la Distribución.

3.3.1 – Gestión del Pase Masivo

Cuando se necesita gestionar un Pase Masivo la Dirección de Transporte UCI solicita al Departamento de Recursos Humanos los datos de las personas a viajar de la UCI y la CUJAE envía un documento con las personas que viajaran por su parte. Con estas listas se obtiene la cantidad de viajeros en total y con este dato la Dirección de Transporte tramita la compra de los pasajes con la Agencia de Viajes que corresponda. Dicha Agencia envía un listado de asientos que son los que corresponden a los pasajes comprados, también vienen los datos referentes a nombre del Transporte, Medio de Transporte, destino, fecha y hora de salida de cada Transporte, en un Pase

Masivo pueden existir varios Transportes que tengan diferente fecha de salida.

Diagrama de Actividades:

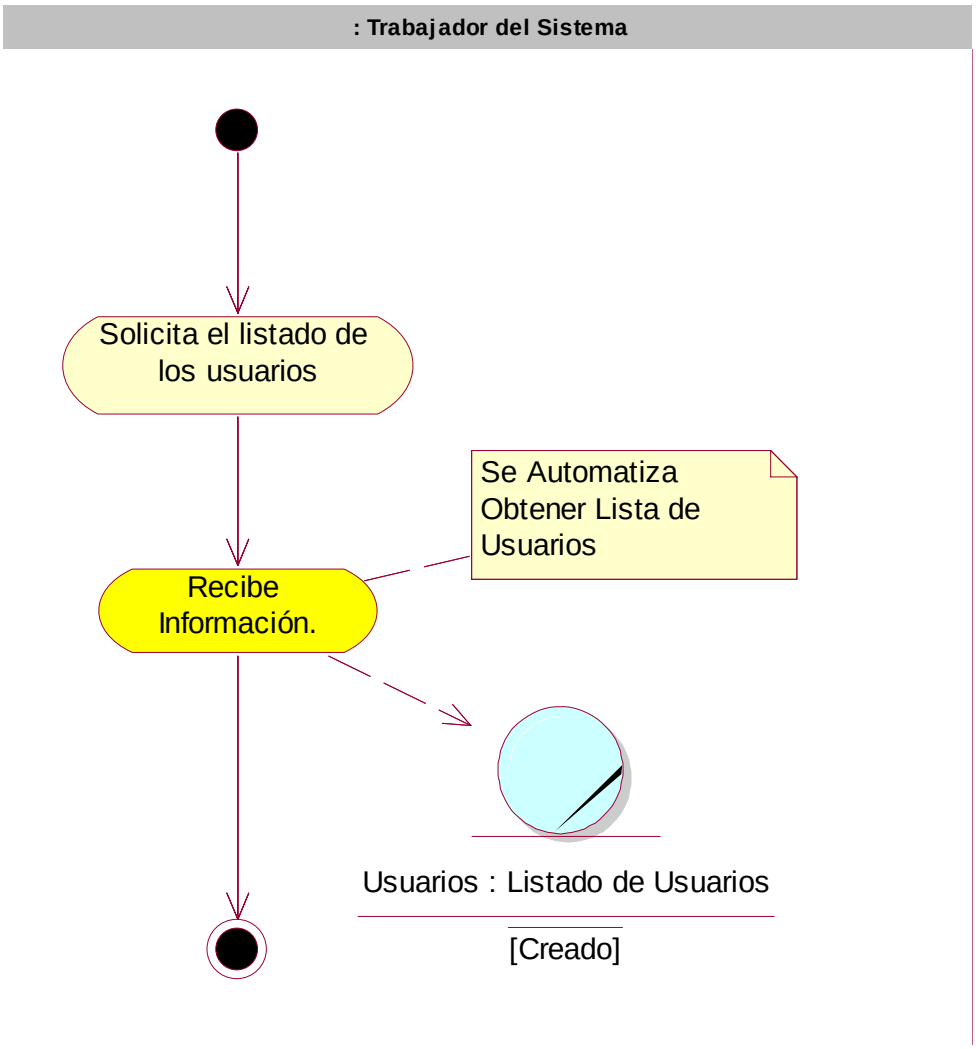


Figura 2. Diagrama de Actividades Gestión Pase Masivo

3.3.2 – Realizar Distribución

Una vez obtenidos todos los datos del transporte la Dirección de Transporte comienza la Distribución según los criterios que deba seguir, cada Pase Masivo cuenta con sus restricciones. En caso de que algún viajero desee

cambiar su destino debe notificarlo a la Dirección de Transporte con antelación y esta hace el cambio de ser posible. Luego de terminada la Distribución se procede a imprimir los Boletines que serán entregados a los viajeros.

Diagrama de Actividades:

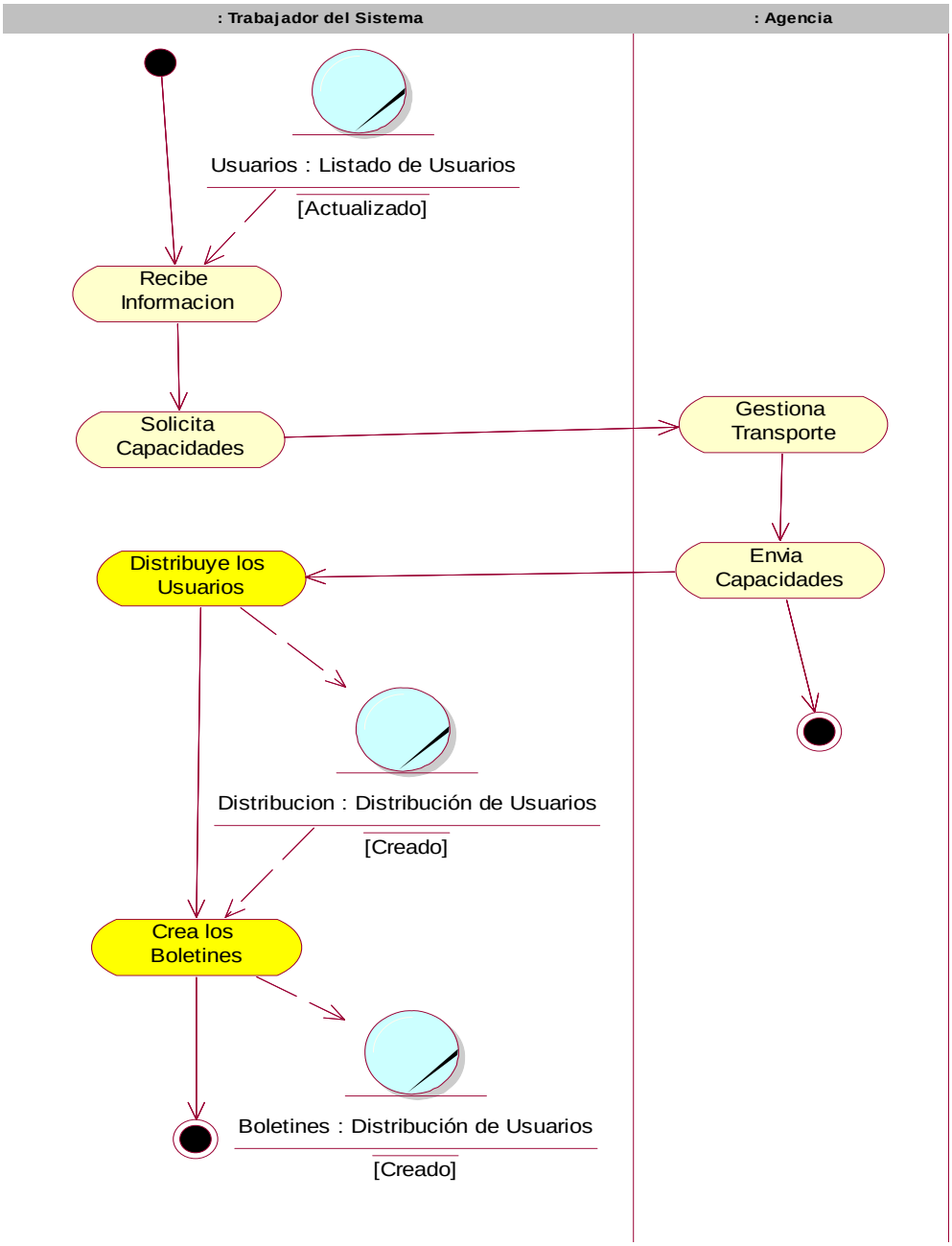


Figura 3. Diagrama de Actividades Realizar distribución

3.4 – Diagrama del Modelo de Objetos del Negocio.

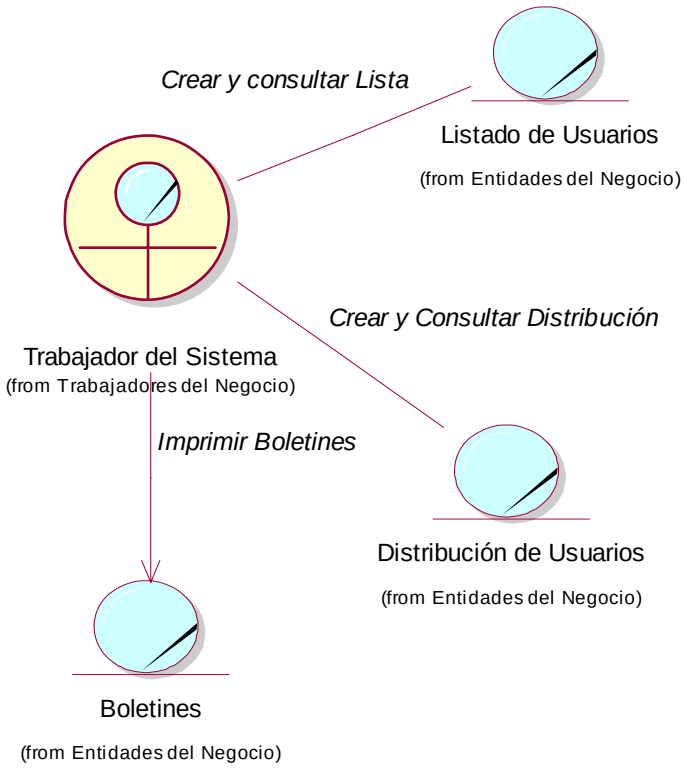


Figura 4. Diagrama del Modelo de Objetos del Negocio

3.5 – Requerimientos Funcionales del sistema

RF1 - Autenticar Usuarios (Autenticación de dominio)

RF2 - Obtener información sobre Viajeros

RF2.1 - Obtener información de un usuario (Nombre/# Carné/usuario UCI/# Identificación UCI)

RF2.2 - Modificar información de un Viajero (Destino/Nombre)

RF2.3 - Obtener ubicación de un Viajero (Nombre/# Carné)

RF3 - Obtener listas con los datos de los Viajeros

RF3.1 - Guardar listas de Viajeros

RF3.2 - Cargar listas de Viajeros anteriormente guardadas

RF3.3 - Obtener lista de Viajeros por criterio (Municipio/Provincia/Tipo Persona)

RF4 - Registrar nuevo Pase Masivo (Nombre, Mes, Año)

RF5 - Registrar nuevo Transporte (Nombre, Destino, Tipo)

RF5.1 - Registrar nuevo Medio de Transporte (Nombre, Destino, Asientos)

RF6 - Crear una Distribución (Criterios de distribución)

RF6.1 - Cambiar Ubicación de uno o varios Viajeros

RF6.2 - Guardar Distribución

RF6.3 - Cargar Distribución previamente guardada

RF6.4 - Eliminar Distribución

RF7 - Editar Diseño de Boletines

RF7.1 - Crear nuevo estilo

RF7.2 - Guardar estilo

RF7.3 - Cargar estilo

RF7.4 - Eliminar estilo

RF8 - Imprimir Boletines

RF8.1 - Generar Documento Protegido (PDF)

RF8.2 - Seleccionar Distribución a Imprimir

RF8.3 - Imprimir selección

3.6 – Requerimientos no Funcionales del sistema

- Interfaz externa

  RNF1 - La interfaz debe ser sencilla, fácil de usar, amigable y mantener el diseño a lo largo de toda la aplicación

  RNF2 - Deberá ser aplicado el estándar de diseño para la interfaz del proyecto Sistema Integral de Reservaciones de la UCI

  RNF3 - Deberán existir dos interfaces, una para los usuarios y otra para los trabajadores que manipulan el sistema.

  RNF4 - Se accederá a toda la aplicación siempre partiendo de la página principal

- Usabilidad

  RNF5 - Debe ser fácil de usar para todo tipo de personas sin importar el nivel de conocimiento que tenga del sistema

- Rendimiento

  RFN6 - Aparte de que en las aplicaciones Web es de suma importancia la rapidez, el sistema debe ser rápido pues trabajará con una gran cantidad de datos que le serán suministrados por diferentes Web Services

  RFN7 - Debe soportar gran cantidad de usuarios conectados al mismo tiempo

- Soporte
  - Extensibilidad

    RNF8 - La aplicación debe ser capaz de aceptar nuevos módulos sin que se afecte su buen funcionamiento
  - Mantenimiento

    RNF9 - La aplicación debe documentarse correctamente de forma tal que el tiempo de reparación, en caso de necesitarse, sea lo mas corto posible
- Seguridad
  - Confiabilidad

    RNF10- El sistema debe implementar varios niveles de seguridad de modo que la información manipulada quede protegida de usuarios no autorizados
  - Integridad

    RNF11- El sistema debe ser capaz de proteger la información y recuperarse de forma rápida de las posibles fallas que ocurran
  - Disponibilidad

    RNF12- Cada usuario debe tener acceso a la información correspondiente a su nivel de acceso en todo momento.
- Requisitos Legales

RNF13- La aplicación y su documentación pertenecen al proyecto Sistema Integral de Reservaciones de la UCI

- Software

  RNF14- Se utilizará como sistema Windows (NT 4.0/ 9.x / 2000/ XP/ 2003 Server), servidor Web el IIS de Windows, servidor de bases de datos MsSql Server 2000 y en al implementación la entorno ASP.NET con el lenguaje C# sobre .NET Framework.

- Restricciones en el diseño y la implementación

  RNF15- Para el diseño se utilizó RUP, con Rational Rose y en la implementación se utilizará Visual Studio .NET y SQL Server como gestor de base de datos.

3.7 – Descripción de la Solución Propuesta

3.7.1 – Descripción de los procesos a automatizar.

Con vistas a resolver el problema que se presenta en la Reservación de Pases Masivos de la UCI se propuso el desarrollo de un sistema que automatizara este proceso. Para ello dicho sistema sería capaz de permitirle al trabajador que lo operaría (Director de Transporte) obtener los datos de las personas a viajar tales como Nombre(s) y Apellido(s), #Carné de Identidad, destino (Municipio), entre otros. Además debe permitir al trabajador del sistema obtener reportes sobre los viajeros en cuanto a cantidad de estos por provincia y municipio o ambos, por tipo, por facultad, grupo y permitir hacer una búsqueda de viajeros y cambiar sus datos si es necesario. Luego debe permitirle entrar los datos correspondientes a los transportes disponibles, Nombre del transporte, Cantidad de medios que lo componen y destino (Provincia), y de sus medios la cantidad de asientos y cuales son, su Nombre y destino (Municipio). Una vez hecho esto el sistema debe aceptar los requisitos dados por el trabajador para realizar la Distribución según los mismos. Además debe permitir hacer una Distribución automática y otra

personalizada. En la automática el sistema solo tendrá en cuenta los requisitos generales, pero en la personalizada es posible emitir requisitos específicos incluso para cada Medio de Transporte y cada Tipo de Persona (estudiantes, profesores, etc.)

Debe proveer además, para uso de los usuarios, un buscador que permita a un usuario saber su ubicación dentro de la Distribución, o sea, saber que asiento y Medio le corresponden.

Por otra parte debe ser posible cambiar los datos de la Distribución por el Trabajador una vez terminada esta, obtener reportes sobre la misma y añadir nuevos viajeros en cualquier momento. Como punto final de la aplicación debe permitir Diseñar el formato de los Boletines e imprimirlos, esta también es una tarea del Trabajador del sistema.

3.7.1.1 - Actores del Sistema

Actores	Descripción
Usuario	Cualquier persona, estudiante, profesor, directivo, familiar o trabajador de la UCI que interactúa con el sistema para obtener una información
Trabajador del Sistema	Persona encargada de usar el sistema para Gestionar las Distribuciones, obtener reportes, distribuir los viajeros, diseñar boletines e imprimirlos
Usuarios del Sistema	Es la generalización que se establece entre los actores Usuario y Trabajador del Sistema, puesto que ambos deben autenticarse en el sistema

3.7.2 – Modelo de Casos de Uso del Sistema

Diagrama de Casos de Uso

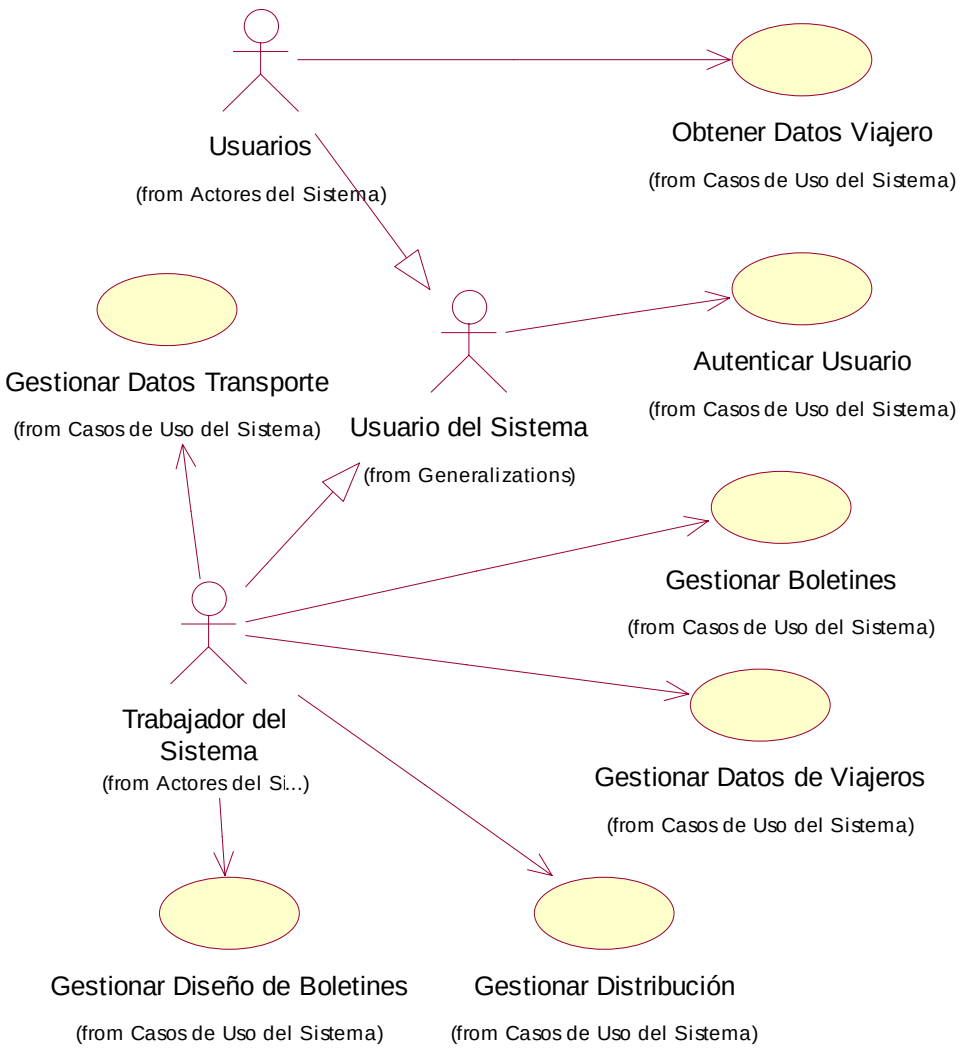


Figura 5. Diagrama de casos de uso del sistema

3.7.3 - Casos de Uso Expandidos

Para lograr una mejor comprensión de los Casos de Uso del sistema a continuación se brinda una descripción más detallada de los mismos, apoyada por las distintas pantallas correspondientes del prototipo que, aunque están sujetas a cambios por el hecho de no estar terminado el diseño final, servirán para facilitar el entendimiento del proceso:

Caso1 de Uso:	**Autenticar Usuario**
Actor(es):	Usuarios del Sistema (Usuario, Trabajador del Sistema)
Propósito:	Identificar el usuario para saber el acceso que tendrá a los recursos del sistema
Resumen:	El Caso de Usa comienza cuando el Actor accede al sistema. En el momento que el Actor solicita conectarse al sistema este obtiene sus credenciales y lo identifica. Una vez identificado el Actor tendrá acceso a la información y utilidades que le correspondan. Este proceso es completamente transparente al Actor puesto que la autenticación es gestionada por el sistema a partir de la autenticación integrada de dominio. En caso de ser un Usuario, cuando intente acceder al sistema verá la página que le mostrara por defecto su Ubicación, si es un Trabajador del Sistema verá la página de distribución principal.

Referencias:	RF1
Precondiciones:	El actor esté autenticado en el dominio
Poscondiciones:	El actor esta autenticado en el sistema por lo que tiene acceso a la información que le corresponde
Requisitos Especiales:	
Pantallas aún en etapa de diseño.	

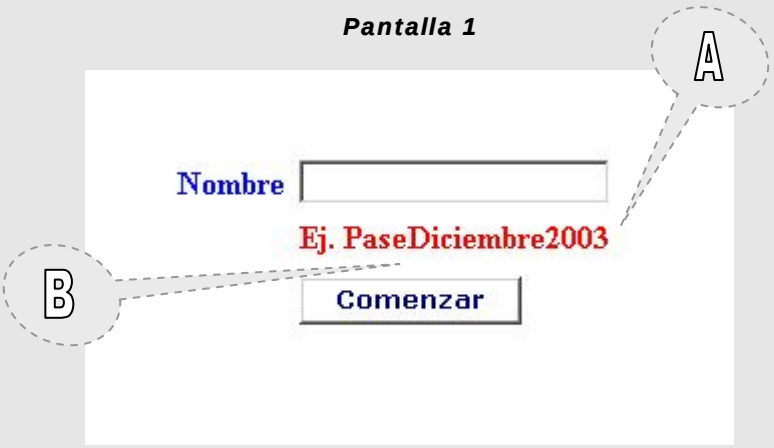
Caso de Uso:	**Obtener Datos Viajeros**
Actor(es):	Usuarios
Propósito:	Obtener determinados datos de un viajero
Resumen:	El usuario introduce un identificador de usuario y el sistema devuelve los datos del usuario referentes a la distribución.
Referencias:	*RF2, RF2.1, RF2*.3
Precondiciones:	Haya sido realizada la distribución
Poscondiciones:	El usuario tiene conocimiento de su ubicación
Requisitos Especiales:	Este usuario solo puede ver los datos referentes a Distribución
Pantallas aún en etapa de diseño.	

Caso de Uso:	**Gestionar Datos de Viajeros**
Actor(es):	Trabajador del Sistema
Propósito:	Obtener los datos de aquellas personas que van a viajar.
Resumen:	Primeramente el actor rea un Nuevo Pase Masivo (*Pantalla 1*) entrando el nombre en caso de estar vacío el nombre el sistema lo advierte y muestra una sugerencia (*Pantalla 1*,*A*), una vez entrado el nombre se procede a comenzar el pase (*Pantalla 1*,*B*), en ese momento el sistema actualiza su base de datos a partir de las bases de datos de la UCI por lo que el proceso puede tomar unos minutos (*Pantalla 2*). Una vez que se han obtenido los datos se conforma la Lista de Viajeros que contiene todos los datos necesarios para realizar la Distribución. Esta lista puede ser conformada de varias maneras según los criterios que el actor decida. Por otra parte el actor puede salvarlas o cargar otras ya salvadas. Ejemplos: Criterio: por tipo de persona (*Pantalla 3*,*A*), tipo de persona combinada con provincia y/o municipio (*Pantalla 3*,*B*). Cantidades: totales por tipo de persona (*Pantalla 3*), de personas por tipo y por provincia (*Pantalla 4 y Pantalla 5*),
Referencias:	*RF3, RF3.1,RF3.2,RF3.3*
Precondiciones:	

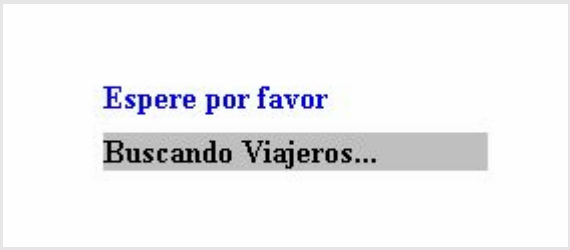
Poscondiciones:	Se obtiene la Lista de datos de los Viajeros según el criterio seleccionado.
Requisitos Especiales:	

Prototipo

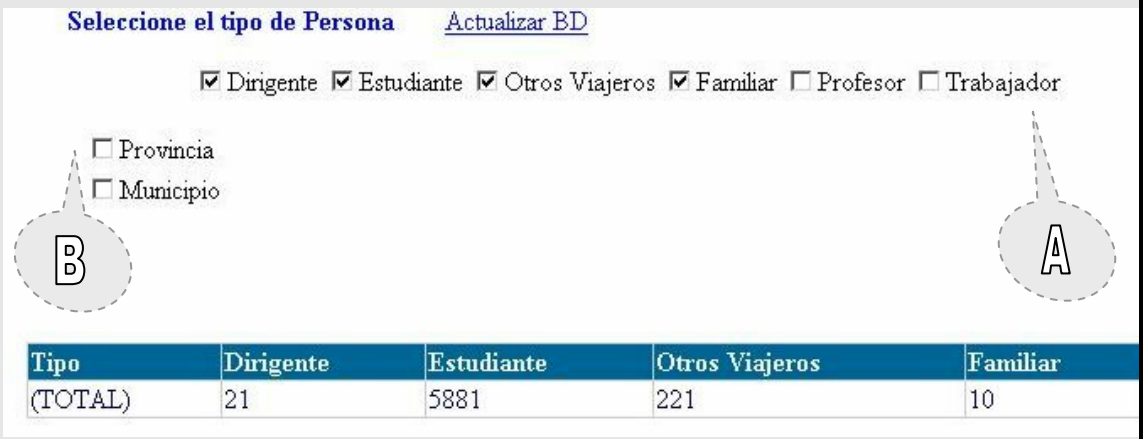
Pantalla 1



Pantalla 2



Pantalla 3



Tipo	Dirigente	Estudiante	Otros Viajeros	Familiar
(TOTAL)	21	5881	221	10

Pantalla 4

Seleccione el tipo de Persona Actualizar BD

☑ Dirigente ☐ Estudiante ☐ Otros Viajeros ☑ Familiar ☐ Profesor ☐ Trabajador

☑ Provincia
☐ Municipio

Provincia	Dirigente	Familiar
Cienfuegos	1	
Ciudad Habana	19	9
Pinar del Río	1	1
(TOTAL)	21	10

Pantalla 5

Seleccione el tipo de Persona Actualizar BD

☑ Dirigente ☐ Estudiante ☐ Otros Viajeros ☑ Familiar ☐ Profesor ☑ Trabajador

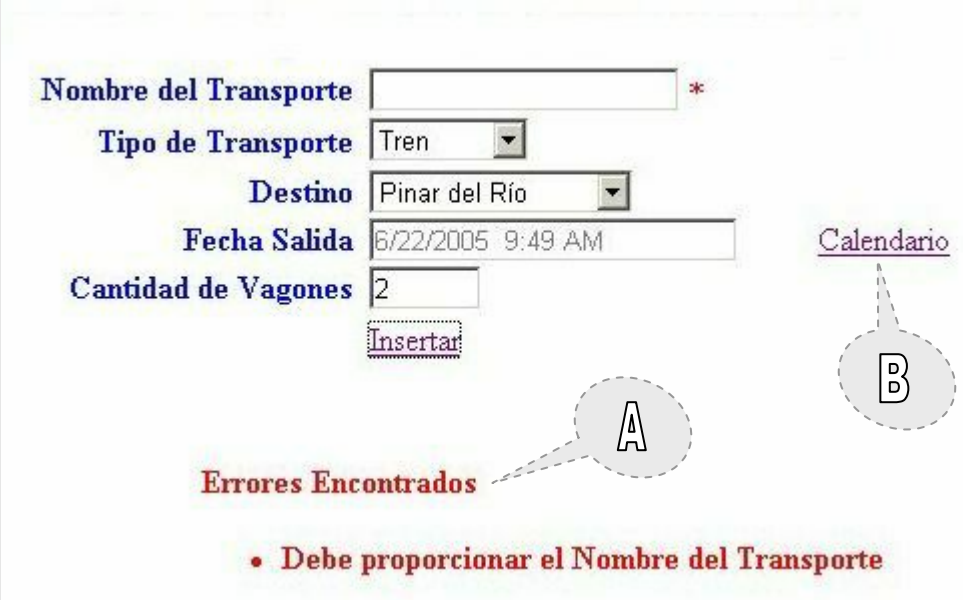
☑ Provincia
☐ Municipio

Provincia	Dirigente	Familiar	Trabajador
Cienfuegos	1		
Ciudad Habana	19	9	12
Pinar del Río	1	1	2
Camagüey			1
Ciego de Avila			2
Granma			1
La Habana			3
Villa Clara			3
(TOTAL)	21	10	24

Caso de Uso:	**Gestionar Datos Transporte**
Actor(es):	Trabajador del Sistema
Propósito:	Guardar los datos referentes a los Transportes a utilizar

Resumen:	A partir de las cantidades de viajeros por provincia obtenidas de la Lista de Datos de Viajeros, el Trabajador del Sistema solicita los transportes y cuando obtiene estos datos procede a incluirlos en el sistema, mediante el formulario correspondiente a los datos de los transportes a utilizar (*Pantalla 1*), donde es necesario darle un nombre al transporte (*Pantalla 2*,*A*) y se inserta la fecha según un calendario (*Pantalla 2*,*B* y *Pantalla 3*). Luego se procede a insertar el transporte y sus correspondientes medios de transporte según el tipo de transporte pulsando el botón correspondiente (*Pantalla 4*,*A*). En esta etapa se insertan los datos y se cuenta con una explicación sobre como entrar los datos de los asientos (*Pantalla 5*) que se muestra en forma de mensaje (*Pantalla 6*) y en caso de ausencia de los datos o error en los mismos se muestran los mensajes requeridos (*Pantalla 7*).
Referencias:	*RF4*,*RF5*,*RF5.1*
Precondiciones:	Lista de Viajeros
Poscondiciones:	Están disponibles todos los datos necesarios para hacer una distribución
Requisitos Especiales:	
Prototipo ***Pantalla 1***	

Pantalla 2

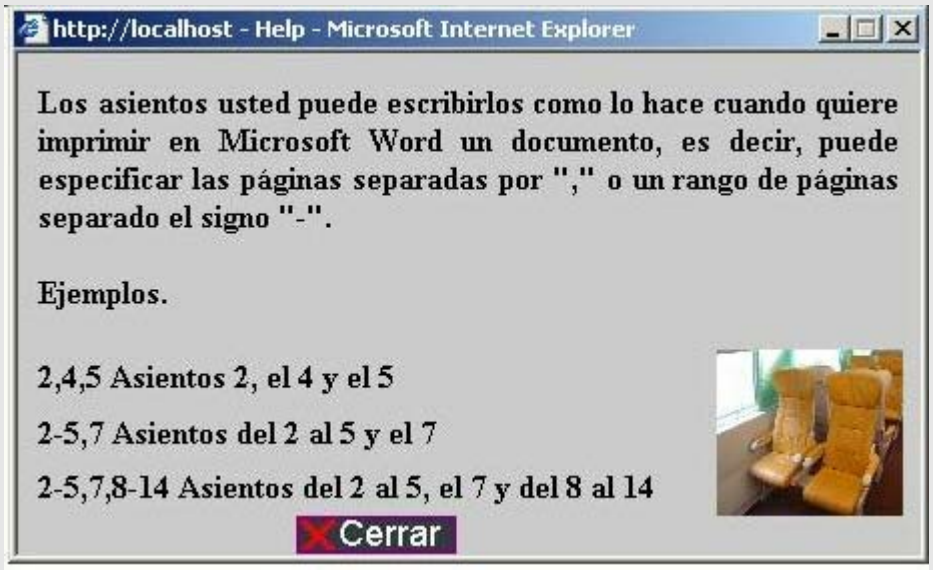


Pantalla 3

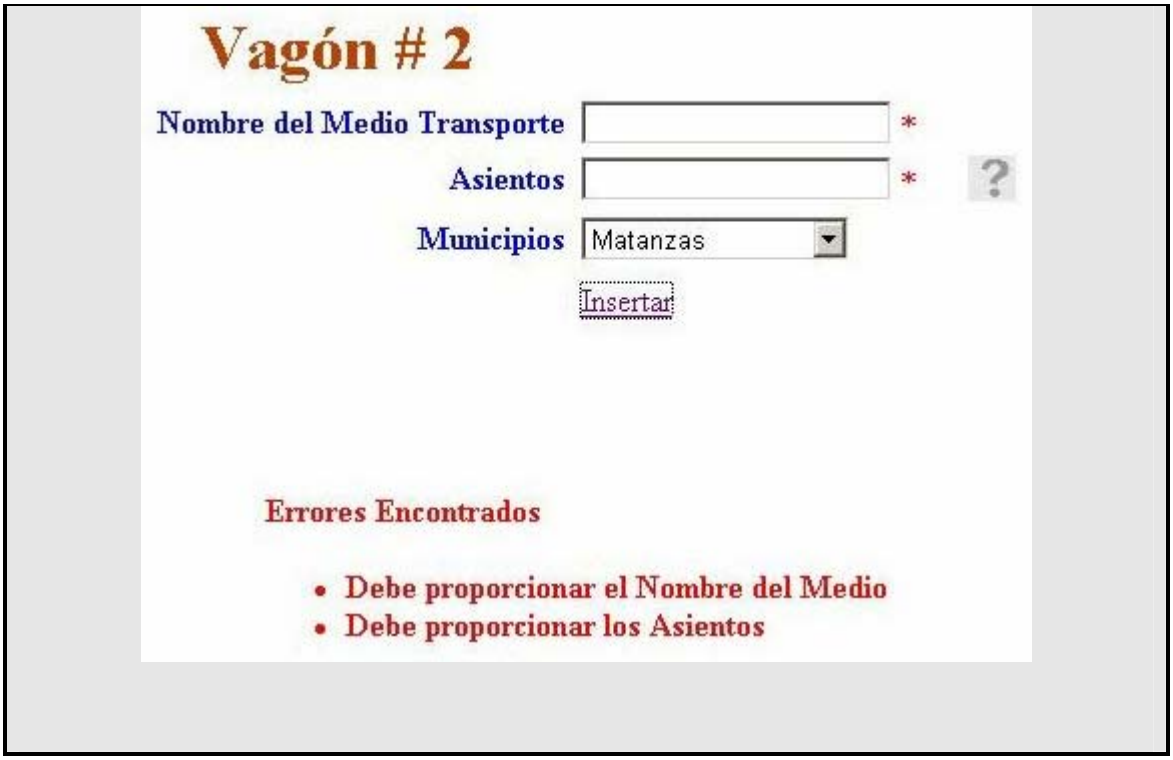
Pantalla 4

Pantalla 5

Pantalla 6



Pantalla 7



Caso de Uso:	**Gestionar Distribución**
Actor(es):	Trabajador del Sistema
Propósito:	Distribuir los Viajeros por los distintos Transportes
Resumen:	El Trabajador introduce los criterios de distribución que debe seguir el sistema, puede escoger también realizar una distribución personalizada. Además le es posible cambiar la ubicación de uno o varios Viajeros, así como guardar o cargar una Distribución.
Referencias:	*RF2.2*, *RF6*, *RF6.1*, *RF6.2*, *RF6.3*, *RF6.4*
Precondiciones:	Lista de Viajeros y Datos de Transporte
Poscondiciones:	Está lista a distribución de los Viajeros

Requisitos Especiales:	
Pantallas aún en etapa de diseño.	

Caso de Uso:	**Gestionar Diseño de Boletines**
Actor(es):	Trabajador del Sistema
Propósito:	Lograr un diseño para los boletines de viaje
Resumen:	El Trabajador puede cargar estilos de boletines ya creados anteriormente o generar nuevos estilos.
Referencias:	*RF7, RF7.1, RF7.2, RF7.3, RF7.4*
Precondiciones:	
Poscondiciones:	Se ha creado el diseño que se le aplicará a los boletines del actual Pase Masivo.
Requisitos Especiales:	
Prototipo	

Caso de Uso:	**Gestionar Boletines**
Actor(es):	Trabajador del Sistema
Propósito:	Imprimir los Boletines

Resumen:	El Trabajador debe seleccionar la distribución que desea imprimir, y esto se guarda en forma de documento protegido y luego es impreso en el momento que se necesite.
Referencias:	*RF8, RF8.1, RF8.2, RF8.3*
Precondiciones:	Diseño de los Boletines y Distribución de Viajeros
Poscondiciones:	Boletines Impresos listos para ser entregados
Requisitos Especiales:	
Pantallas aún en etapa de diseño.	

3.8 - Conclusiones

En el presente capítulo se ha desarrollado la propuesta de solución, obteniéndose los casos de uso del sistema y las funciones que debe tener el mismo partiendo de un análisis profundo de los procesos del negocio. Basándose en los requerimientos y funciones que han sido considerados a lo largo del capitulo se comienza a elaborar la solución al problema. Además se ha descrito la forma en que el sistema dará solución al problema en cuestión.

Capítulo 4

Implementación de la solución propuesta

4.1 - Introducción

En el presente capítulo se modela todo lo necesario para la correcta construcción de la aplicación Web que se propone. Con el fin de lograr un mejor entendimiento y fluidez han sido separados los componentes de la aplicación en paquetes según su funcionalidad, presentándolos como clases en los diagramas de clases Web. También se aborda el modelo de datos utilizado para la construcción de la base de datos y finalmente se exponen los principios de diseño del sistema.

4.2 - Diagrama de Clases

La solución propuesta, de acuerdo con la arquitectura de capas en que se basa el diseño, está formada por tres proyectos:
gaAccesoDatos: Se encarga de la Capa de Acceso a Datos
gaReportes: Encargado de elaborar los reportes necesarios.
PaseMasivo: Implementa las Clases encargadas del proceso principal del negocio y la edición de los boletines, además de contener la Interfaz Web.

Para lograr una mayor claridad y comprensión de las clases del sistema se ha dividido la modelación en paquetes y subpaquetes según sus funcionalidades respectivas:

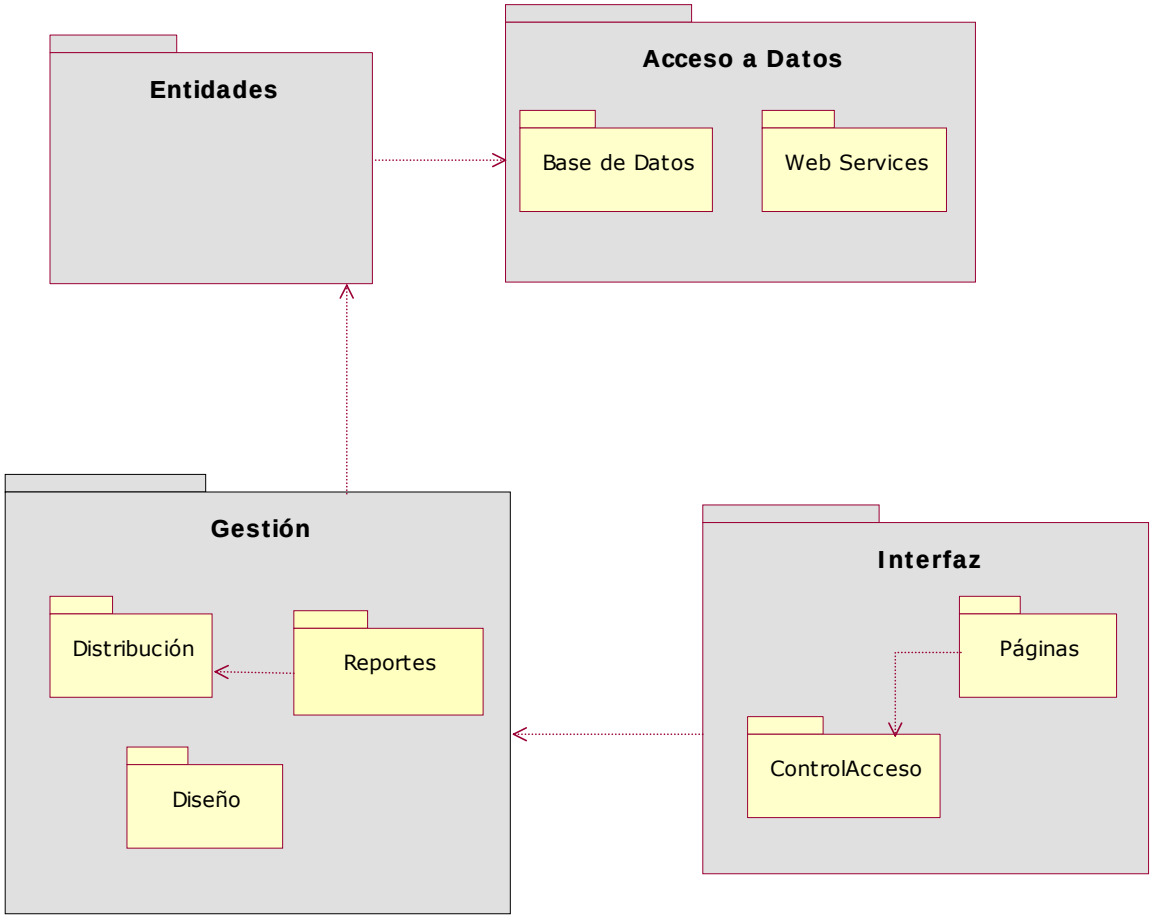


Figura 6. Diagrama de Clases

La capa o paquete de *Acceso a Datos* engloba los subpaquetes *Base de Datos* y *Web Services* que a su vez agrupan las clases encargadas de gestionar las conexiones a la base de datos SQL y a los distintos Web Services respectivamente. Estas clases son las encargadas de la gestión de todos los datos necesarios para el buen funcionamiento de la aplicación y de preservar la integridad y fidelidad de dichos datos.

El paquete *Entidades* contiene las clases que representan las entidades físicas de la base de datos, o sea son las encargadas de la gestión de los datos utilizando la capa de *Acceso a Datos*.

El paquete *Gestión* agrupa las clases encargadas del procesamiento de la información con que trabaja el sistema y de dar solución al problema que se presenta. Este paquete contiene los subpaquetes *Diseño*, *Distribución* y *Reportes.* El subpaquete *Distribución* reúne las clases que se encargan de la manipulación de información, realización de la Distribución de Viajeros y manejo de datos en general, mientras que el de *Reportes* contiene las clases encargadas de elaborar los distintos reportes que el usuario o cliente necesiten. Por su parte el de *Diseño* comprende las clases necesarias para la elaboración y manipulación de los boletines de viaje.

El paquete *Interfaz* esta formado por los subpaquetes *Páginas*, contiene las clases con las que interactuará el usuario o sea la interfaz Web y *ControlAcceso* contiene la clase que se encarga de controlar el acceso de los usuarios a las diferentes páginas.

4.2.1 - Paquete Acceso a Datos

4.2.1.1 - Subpaquete Base de Datos

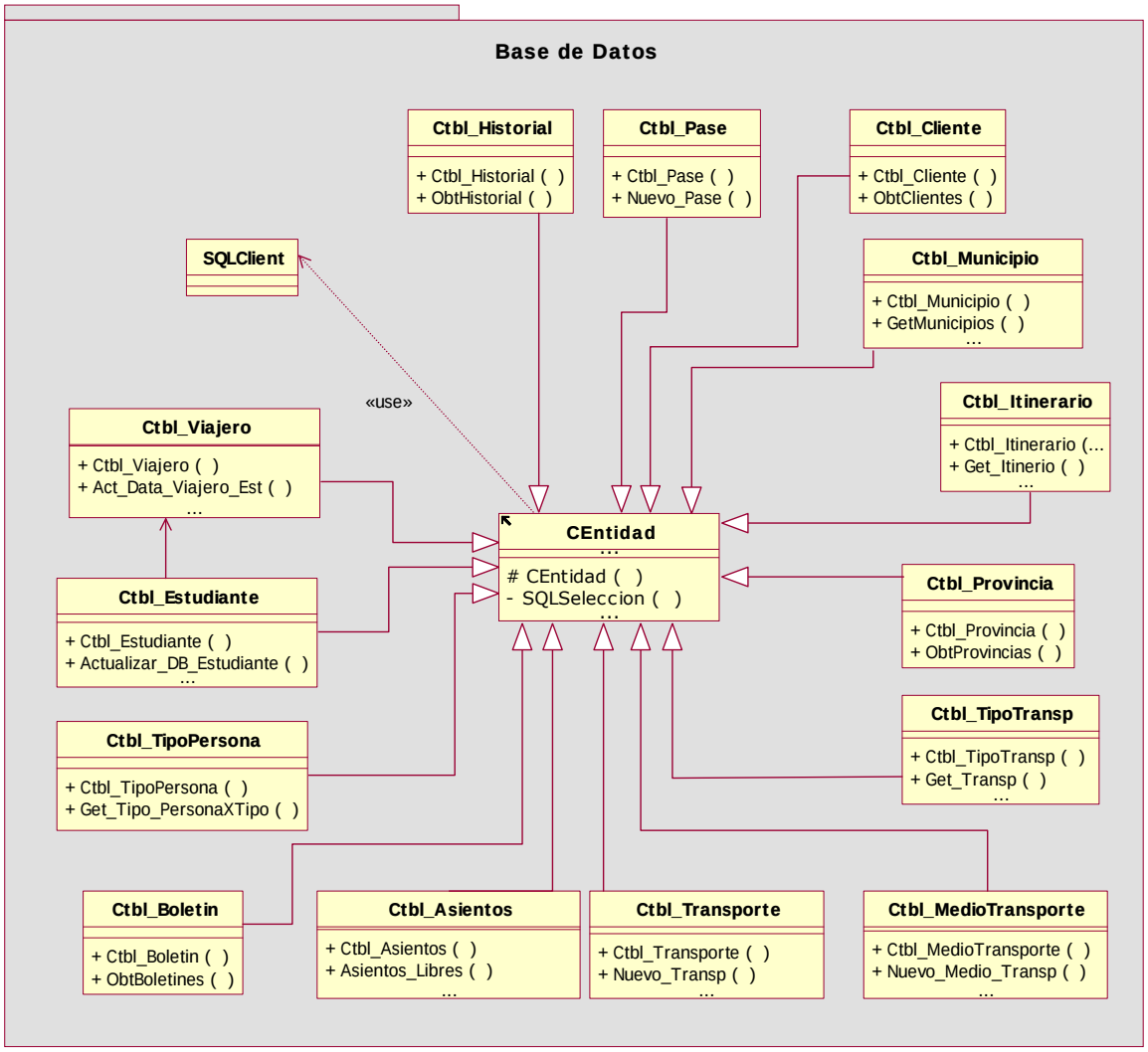


Figura 7. Diagrama de Clases. Subpaquete Base de Datos.

4.2.1.2 - Subpaquete Web Services

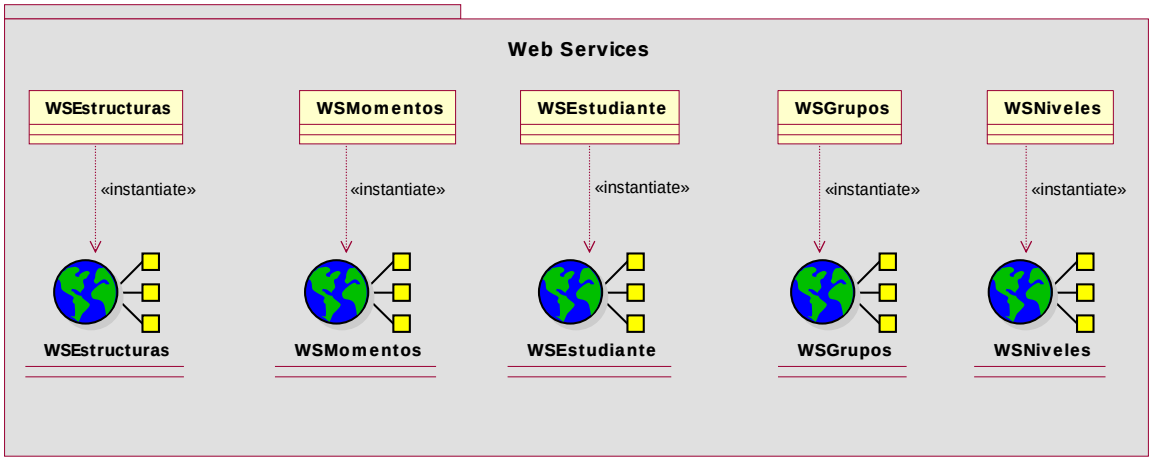


Figura 8. Diagrama de Clases. Subpaquete Web

4.2.2 - Paquete Gestión

4.2.2.1- Subpaquete Distribución

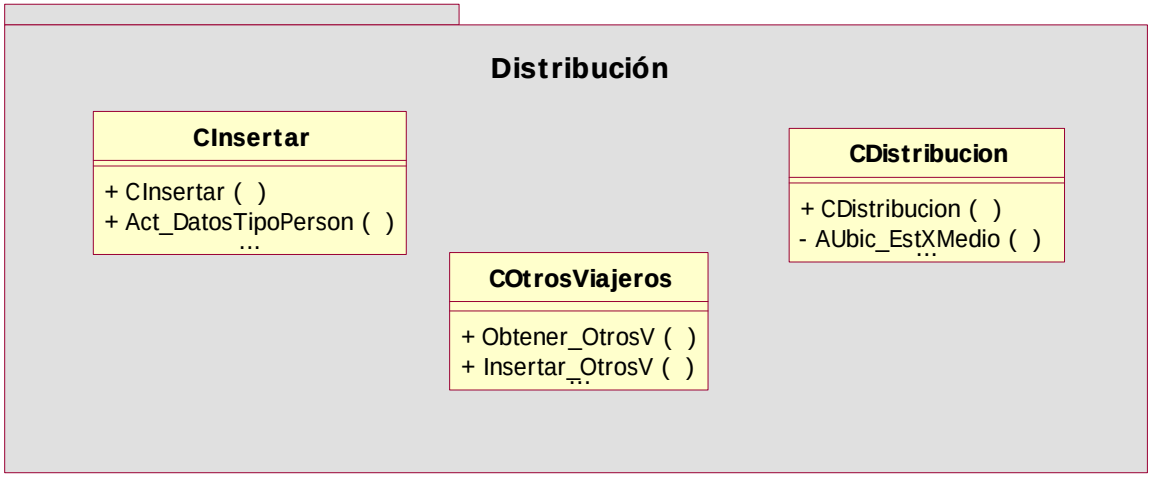


Figura 9. Diagrama de Clases. Subpaquete Distribución.

4.2.2.2 - Subpaquete Reportes

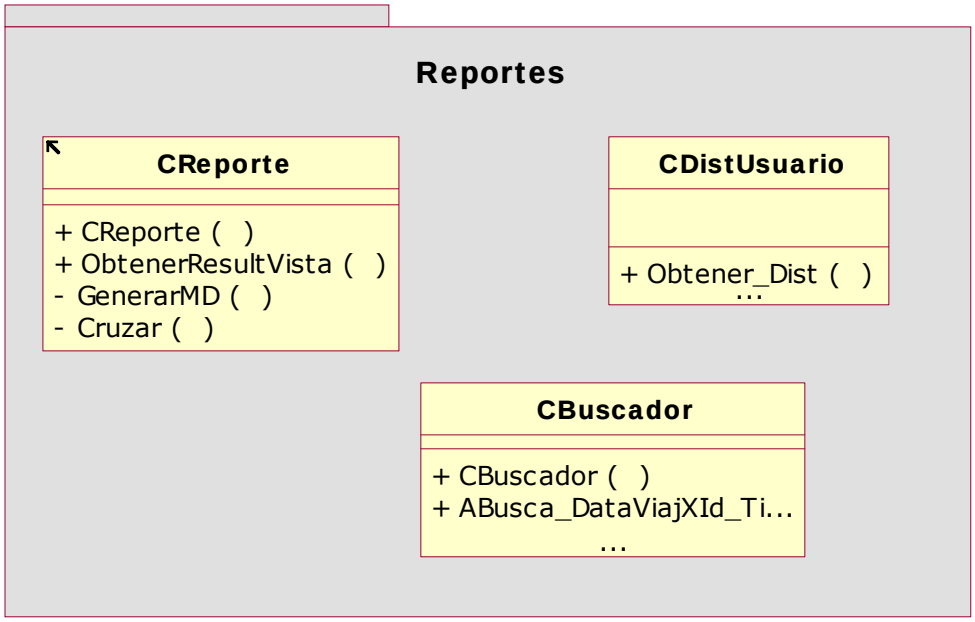


Figura 10. Diagrama de Clases. Subpaquete Reportes.

4.2.3 - Paquete Interfaz

4.2.3.1 - Subpaquete Páginas

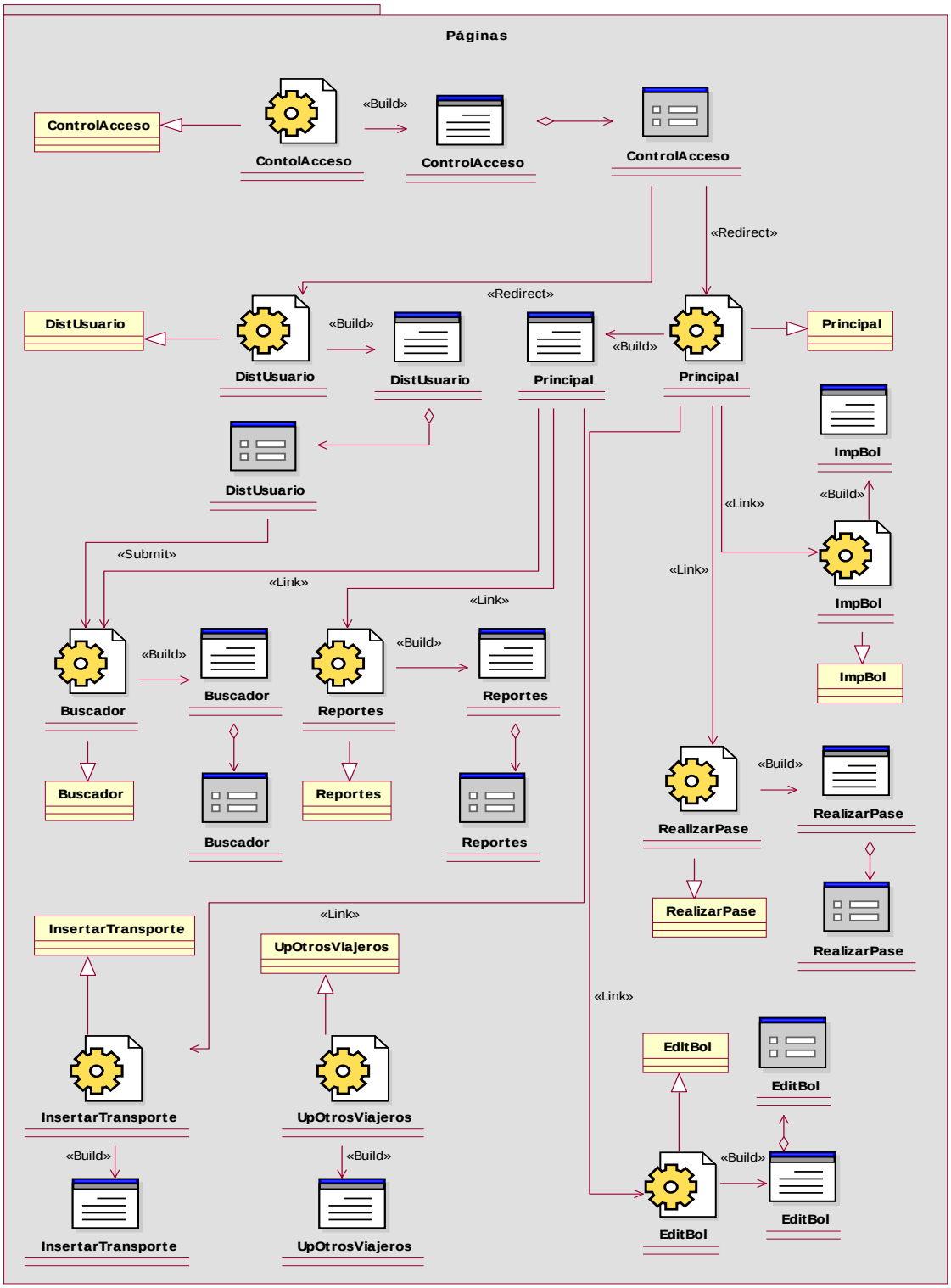


Figura 11. Diagrama de Clases. Subpaquete Páginas.

4.2.3.2 - Subpaquete ControlAcceso

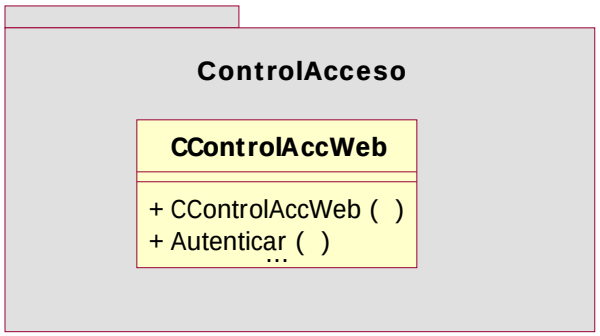


Figura 12. Diagrama de Clases. Subpaquete ControlAcceso

4.2.4 - Paquete Entidades

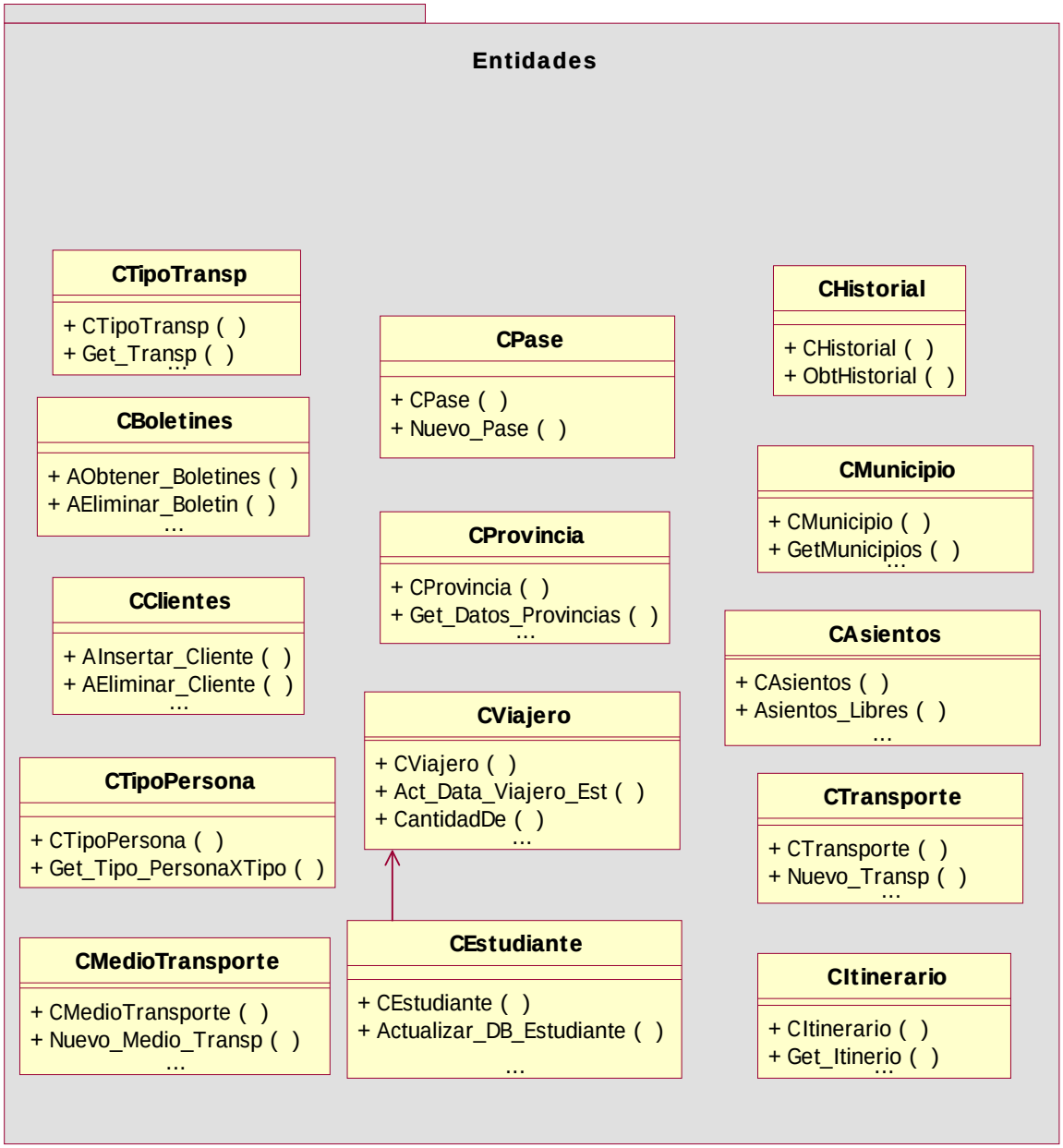


Figura 13. Diagrama de Clases. Paquete Entidades

4.3 - Diseño de la Base de Datos

En la construcción de la base de datos se partió del diagrama de clases persistentes para obtener el modelo de datos del sistema, el cual es completamente compatible con la base de datos.

4.3.1 - Diagrama de Clases Persistentes

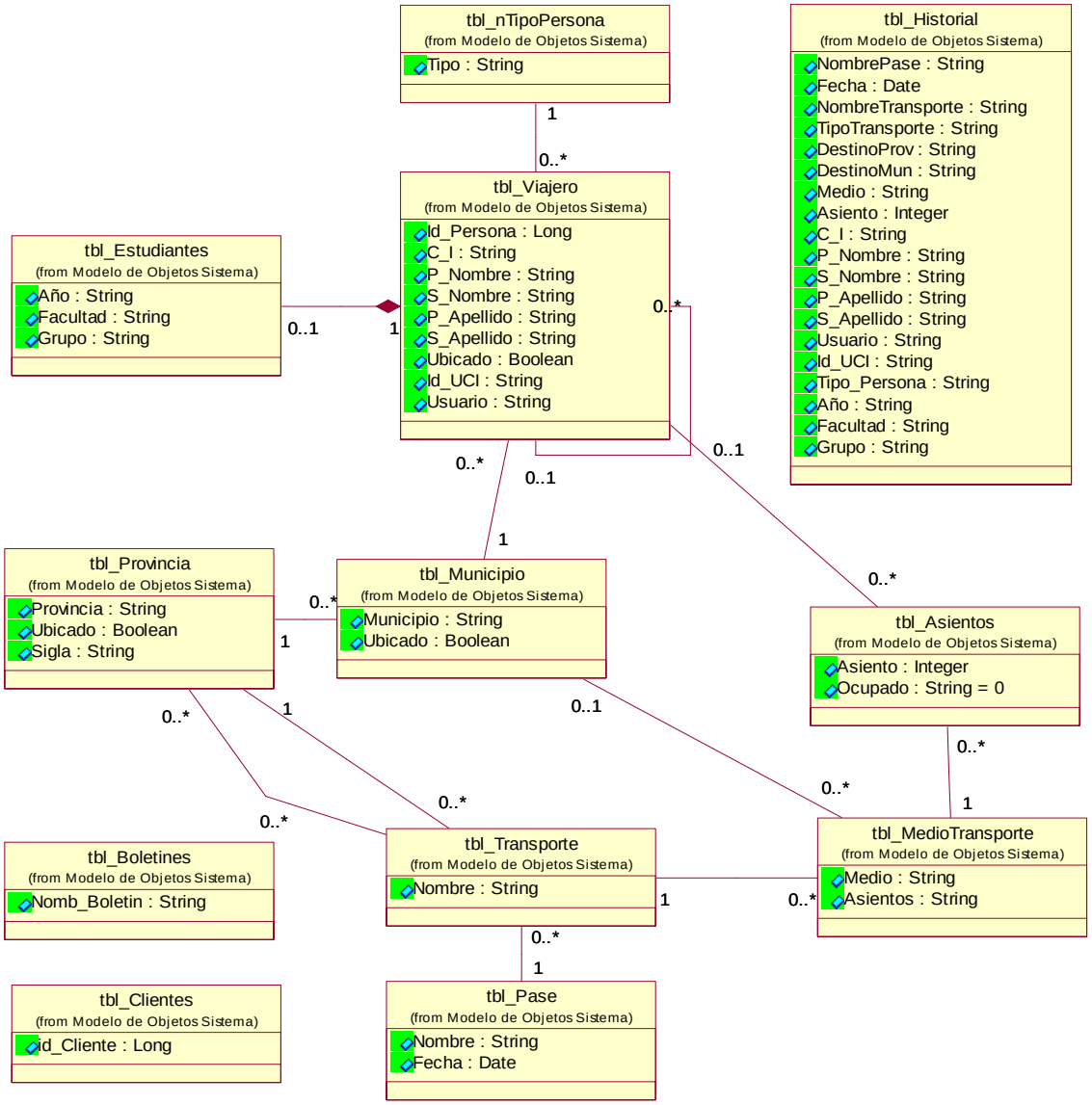


Figura 14. Diagrama de Clases Persistentes.

4.3.2 - Modelo de datos

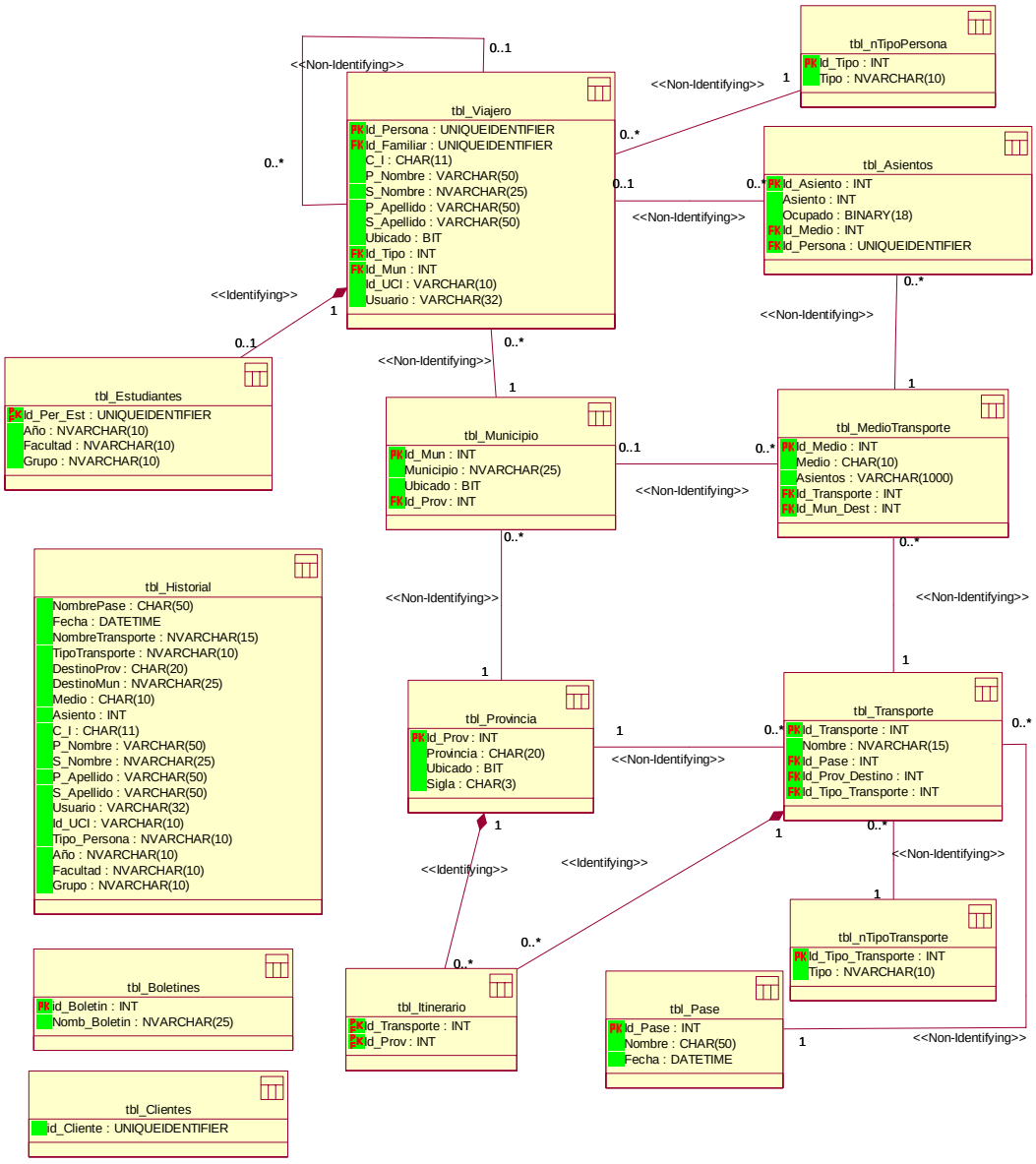


Figura 15. Diagrama del Modelo de Datos.

4.4 - Principios de diseño

El diseño trata de mantenerse en toda la aplicación para lograr que sea fácil acostumbrarse al sistema y lograr una identificación. Además está basado en una interfaz amigable, sencilla y fácil de comprender ya que los usuarios no necesariamente tienen que contar con conocimientos informáticos.

4.4.1 - Estándares de la Interfaz de la Aplicación

Como módulo del Proyecto Sistema Integral de Reservaciones de la UCI el sistema "heredará" el formato de este, solo que implementará sus propios controles. Las páginas Web tienen una estructura basada en plantillas donde se identifica el sistema y están los controles comunes a todos los módulos. A continuación se muestra el formato base de las plantillas que se utilizan:

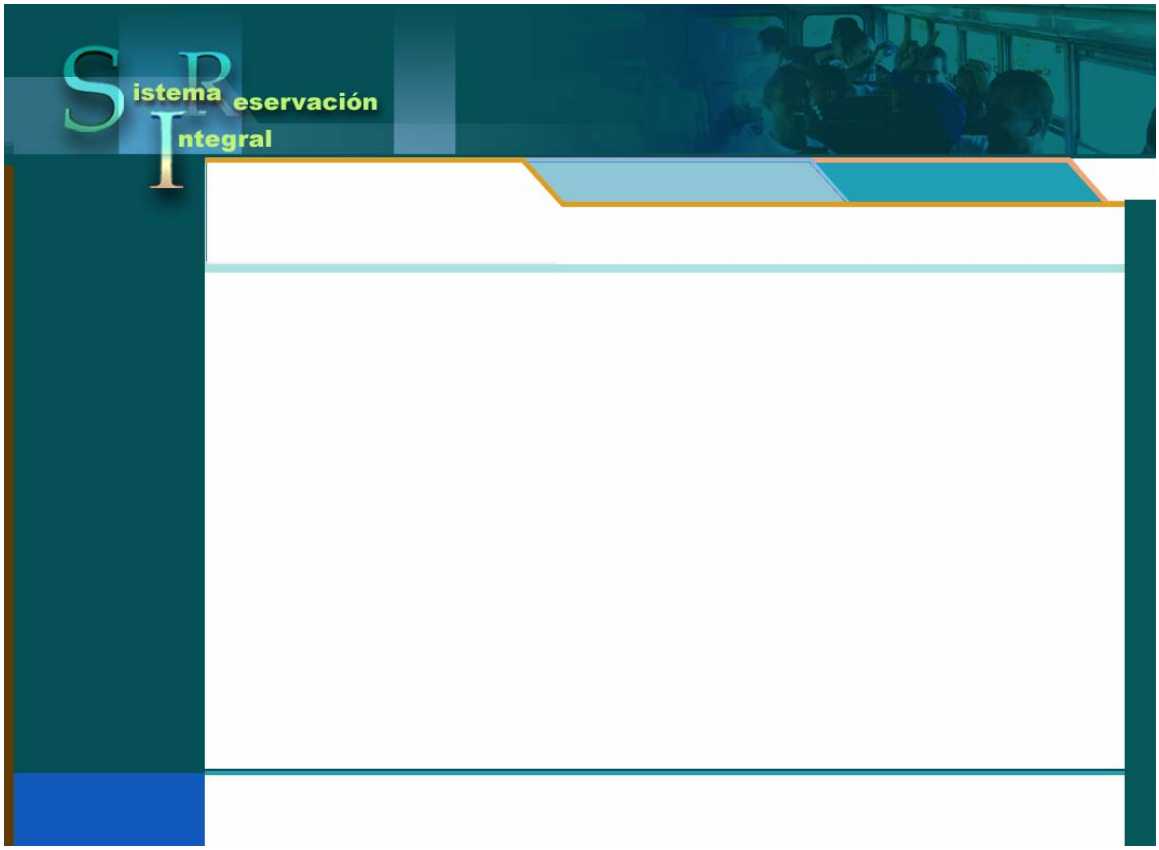


Figura 16. Imagen en la que se basa es diseño de las plantillas

Como se puede observar se utilizan colores en su mayoría con tonos de gris, azul, verde y negro. Además se ha tenido especial cuidado en no sobrecargar las páginas con banners y controles innecesarios, tratando siempre de mantener un formato claro y lo más sencillo posible sin perder funcionalidad.

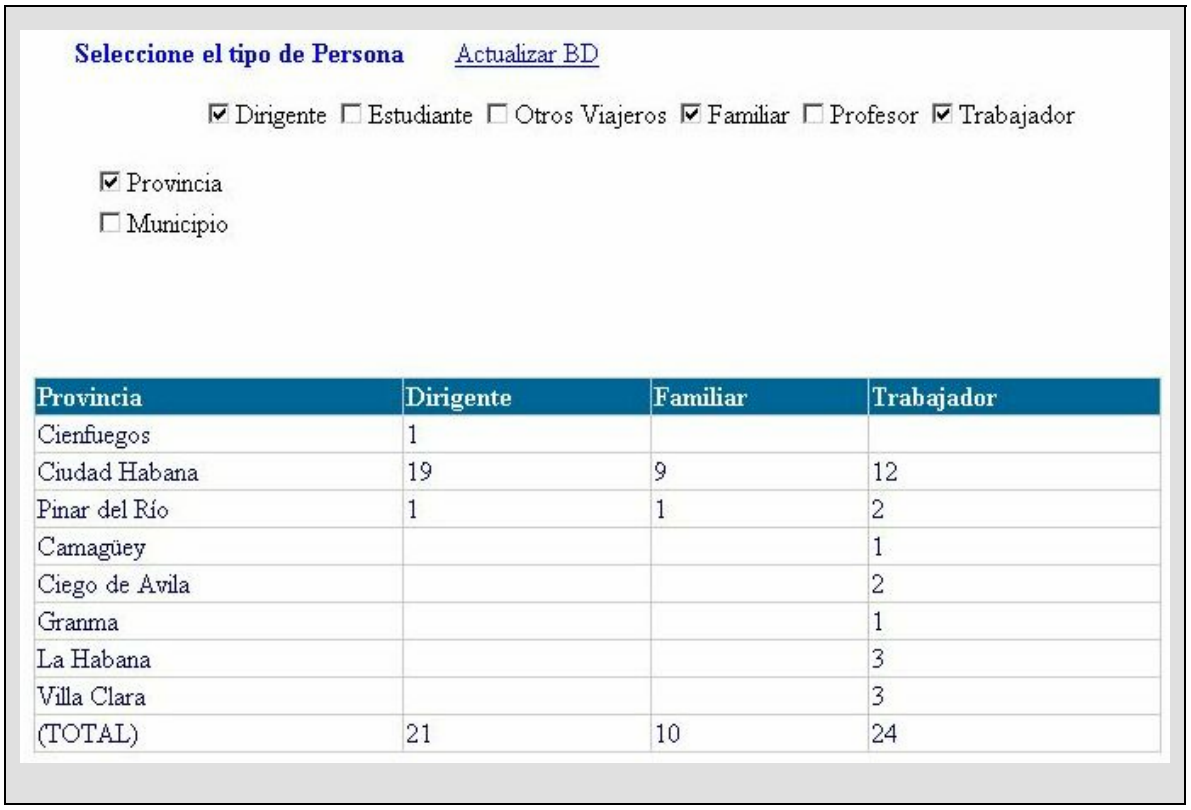
El color rojo esta presente en los mensajes de errores de campos requeridos o con formato incorrecto como se muestra seguidamente:

Figura 17. Color rojo para mostrar errores u omisiones

En general se realizan múltiples operaciones en cada página, de forma que el usuario no tenga que moverse tanto dentro de la aplicación, para completar una operación. Por ejemplo, se puede hacer la inserción, actualización y eliminación de viajeros en las páginas donde se muestran los listados.

4.4.2 - Formatos de Reportes

El sistema brinda varios reportes sobre las diferentes informaciones que utiliza o procesa. Los reportes pueden estar en forma de listas o de un solo elemento. Las listas se pueden ordenar por cualquier campo para facilitar la búsqueda de información. Se decidió no admitir paginado pues aunque la lista puede ser larga, su principal objetivo es la impresión de estos datos. A continuación se muestra un ejemplo de reporte:



Provincia	Dirigente	Familiar	Trabajador
Cienfuegos	1		
Ciudad Habana	19	9	12
Pinar del Río	1	1	2
Camagüey			1
Ciego de Avila			2
Granma			1
La Habana			3
Villa Clara			3
(TOTAL)	21	10	24

Figura 18. Ejemplo de reporte

4.4.3 - Concepción general de la Ayuda

El sistema esta diseñado de forma sencilla y sobre la base de que los trabajadores del sistema, que son los que mayor interacción con el mismo, conocen su trabajo. Es por ello que no se cree necesaria una ayuda de gran magnitud por lo que solo se implementa una ayuda basada en los "hints" o sugerencias. Sin embargo si es necesario el sistema implementa ayuda en forma de mensaje sobre el formato de los datos que deben entrarse. Ejemplo:

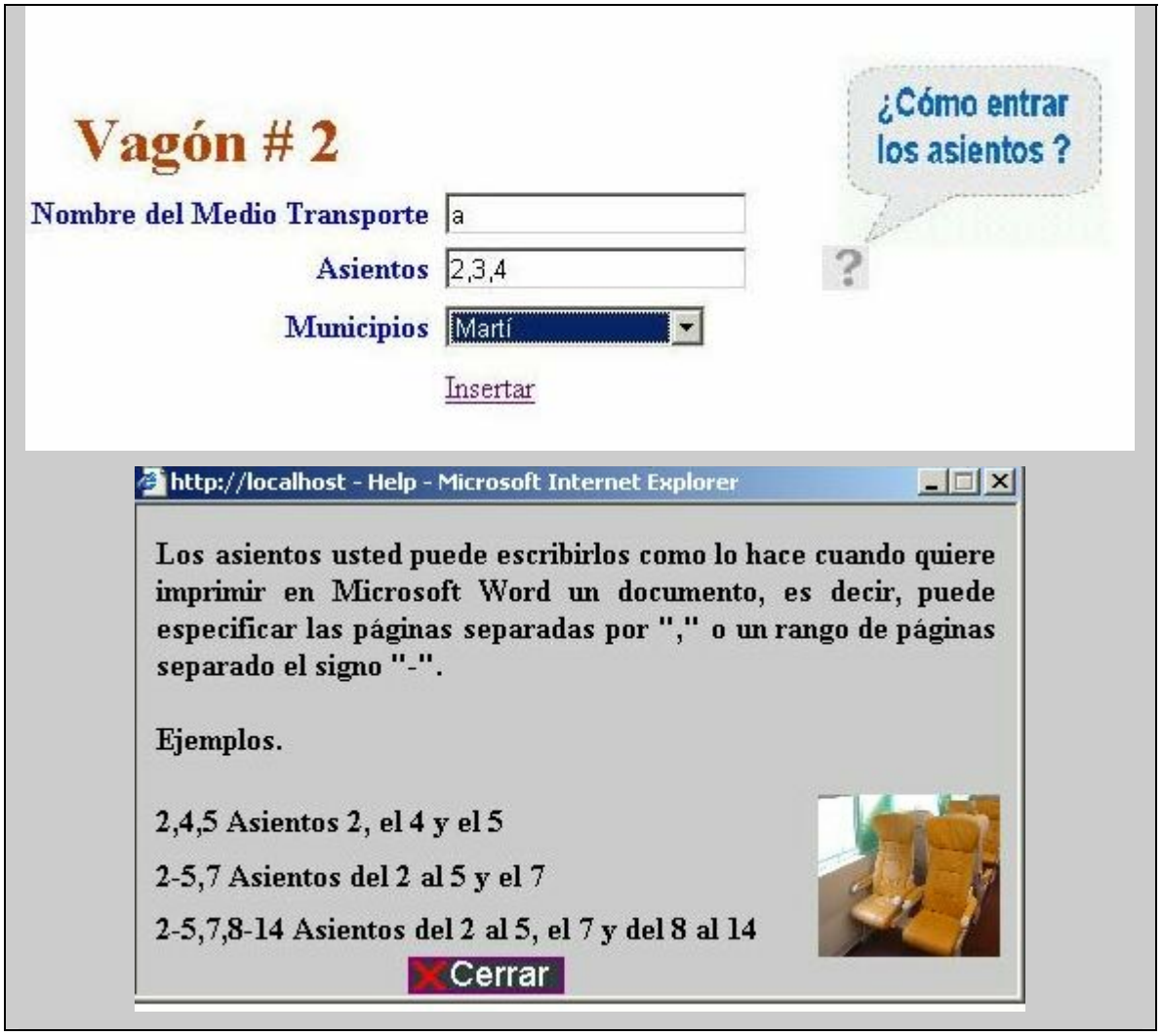


Figura 19. Ejemplo de ayuda

4.4.4 - Tratamiento de excepciones

Para prevenir errores por parte del usuario, sólo se le brindan las opciones mínimas necesarias, a la hora de efectuar cualquier operación, por ejemplo, se deshabilitan ciertos botones si el usuario no tiene que utilizarlos en ese momento.

Mediante una combinación de validación en el lado del cliente y en el lado del servidor, se garantiza que los datos suministrados por los usuarios, se almacenen íntegros y no existan inconsistencias. Se verifican los campos obligatorios, y se revisa el tipo de datos.

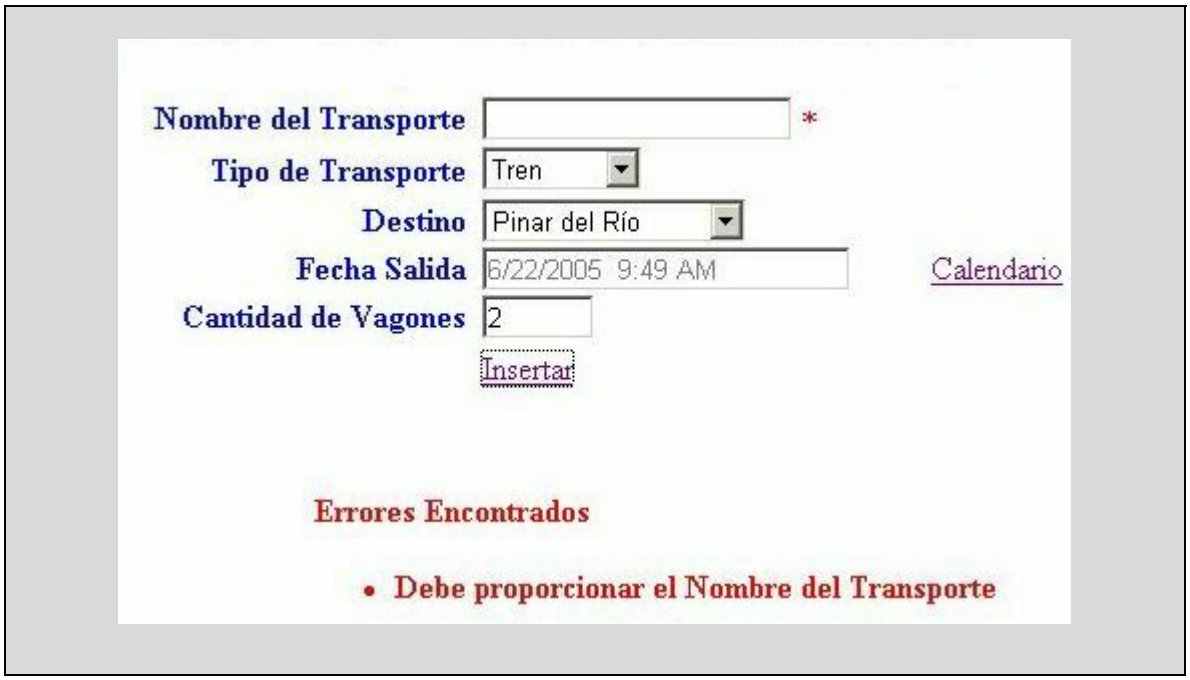


Figura 20. Verificación de campos requeridos y formato correcto

4.5 - Estándares de codificación

La utilización de un estándar de codificación trae ventajas como la reducción de errores, el código se muestra comprensible y claro, se garantiza la comunicación entre los programadores y el mantenimiento del software es más sencillo y rápido. En esta aplicación se ha seguido el estilo de codificación propuesto por Microsoft para programar con C#. Se tomó como estándar nombrar las clases, variables u otros elementos agregándole al nombre un prefijo que represente su tipo. Quedando, por ejemplo, de esta forma:

- Campos de edición de texto: **Txt**NombreCampo.
- Clases: **C**NombreClase
- Letreros: **Lbl**NombreLabel
- CheckBox: **ChkBx**NombreBox
- Botones: **But**NombreBoton.
- Variables de control de ciclos: **i**, **j**, **k**
- Datagrids: **DG**NombreDataGrid
- Datasets: **ds**NombreDataSet
- Datarows: **dr**NombreDataRow
- DataTables: **dt**NombreDataTable

Estos han sido solo algunos ejemplos pero sirve para dar una idea general de el estándar de nomenclatura de los elementos utilizado

Por último y no menos importante a la hora del diseño de la base de datos, las tablas se han nombrado igual que la entidad que almacenan, pero anteponiéndoles el prefijo 'tbl_', o sea si la entidad es Casa, la tabla sería tbl_Casa. Los campos de estas tablas están nombrados igual que las propiedades de las entidades.

4.6 - Modelo de despliegue

El modelo de despliegue muestra la forma en que las distintas partes que conforman la aplicación están distribuidas físicamente. Es decir muestra los componentes físicos, dígase servidores y demás elementos, sobre los que se ejecuta la aplicación.

Para la implementación del sistema se ha seguido la arquitectura de tres capas, donde se separan los componentes de Acceso a Datos, del Cliente y la Lógica del Negocio.

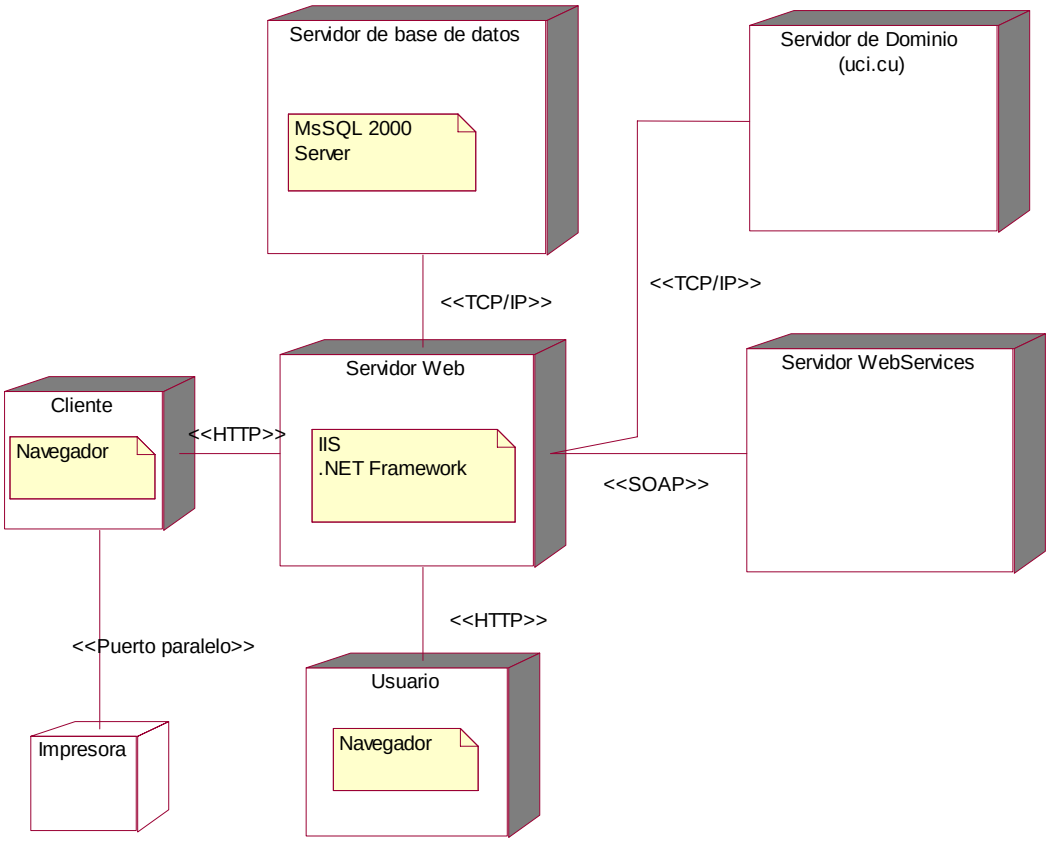


Figura 21. Diagrama de Despliegue.

4.7 - Modelo de Implementación

4.7.1 - Componentes de los diagramas

Con la finalidad de lograr una explicación clara y comprensible de los componentes de la aplicación se ha decidido mantenerlos agrupados por paquetes según su propósito y funcionalidad.

Paquete Base de Datos:

Contiene los componentes que implementan las clases de acceso a la base de datos en SQL Server y obtienen los datos que resulten necesarios.

Paquete Web Services:

Contiene los componentes encargados de conectarse a los diferentes Web Services que brindan los datos necesarios de personas, estudiantes, trabajadores, etc.

Paquete Entidades:

Contiene los componentes que implementan las clases que representan las clases persistentes, o sea son las encargadas de asegurar la persistencia de los datos y su integridad.

Paquete Distribución:

Recoge los componentes que implementan las clases que procesan la información obtenida para obtener la distribución consecuentemente con los criterios dados.

Paquete Reportes:

Componentes que contienen las clases que generan todos los reportes que se necesitan en el proceso.

Paquete Interfaz:

Reúne todos los componentes que implementan las clases que sirven de interfaz a la aplicación. Estos son los encargados de mostrar los resultados y de repcionar los datos que se requieran. Debido a la plataforma .NET cada componente esta dividido en dos: Código HTML y Código CS. Además contiene los componentes necesarios para el control de usuarios y seguridad del sistema.

4.7.2 - Diagrama de Componentes

- Paquete Base de Datos

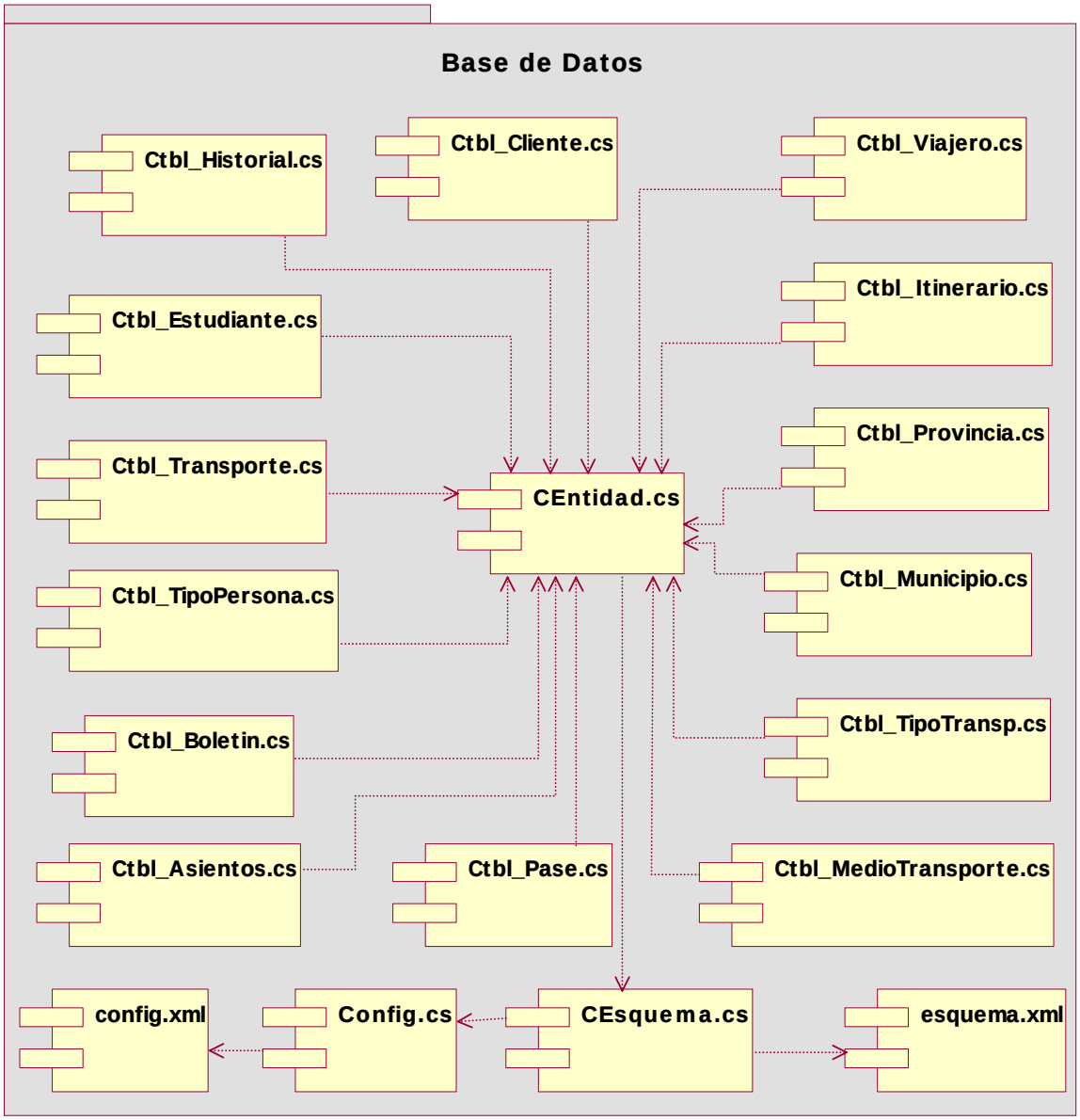


Figura 22. Diagrama de Componentes. Paquete Base de Datos.

- Paquete Web Services

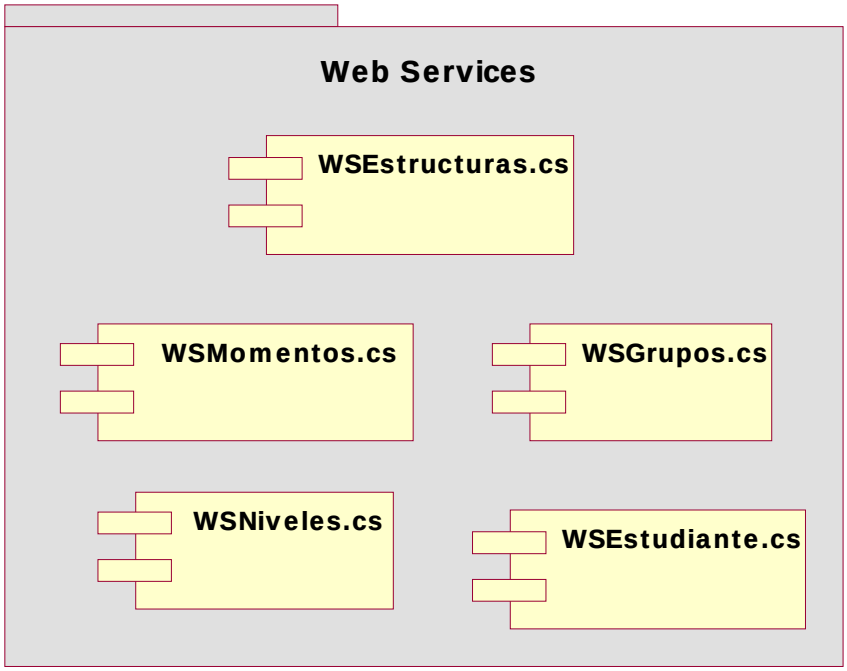


Figura 23. Diagrama de Componentes. Paquete Web Services.

- Paquete Reportes

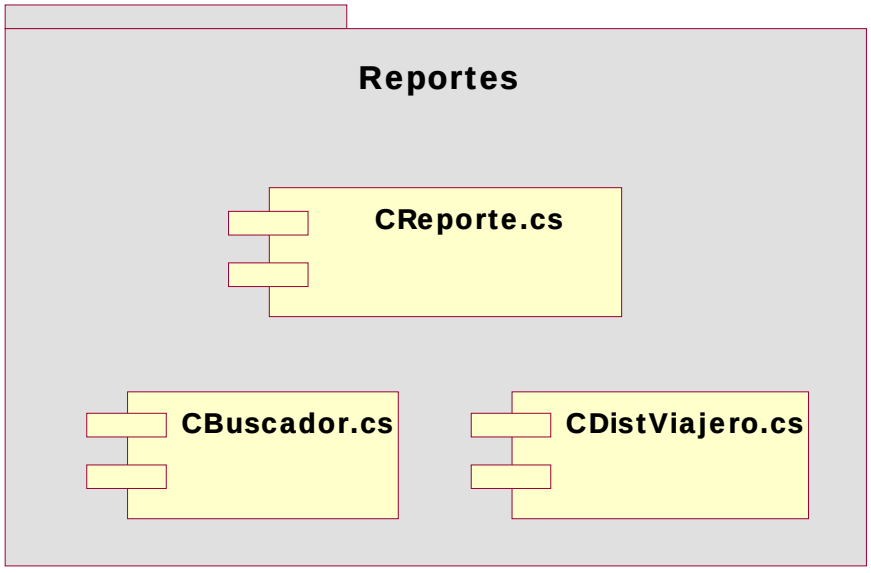


Figura 24. Diagrama de Componentes. Paquete Reportes.

- Paquete Distribución

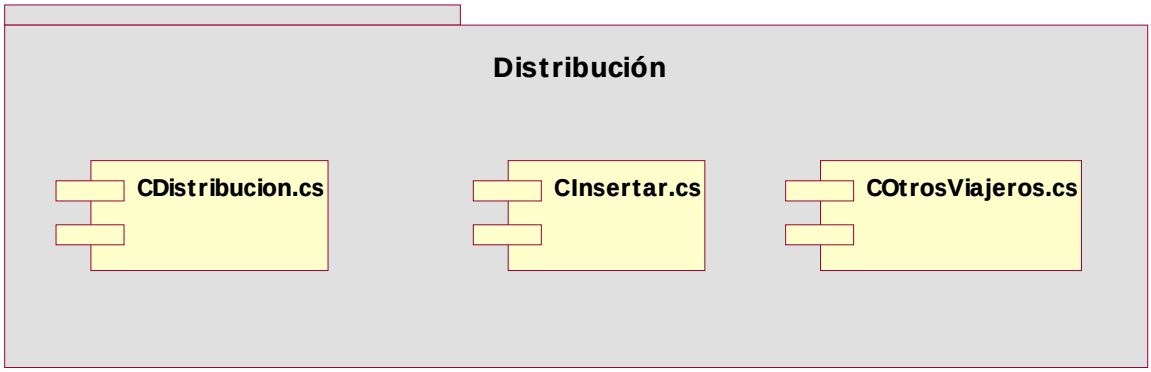


Figura 25. Diagrama de Componentes. Paquete Base de Datos.

- Paquete Entidades

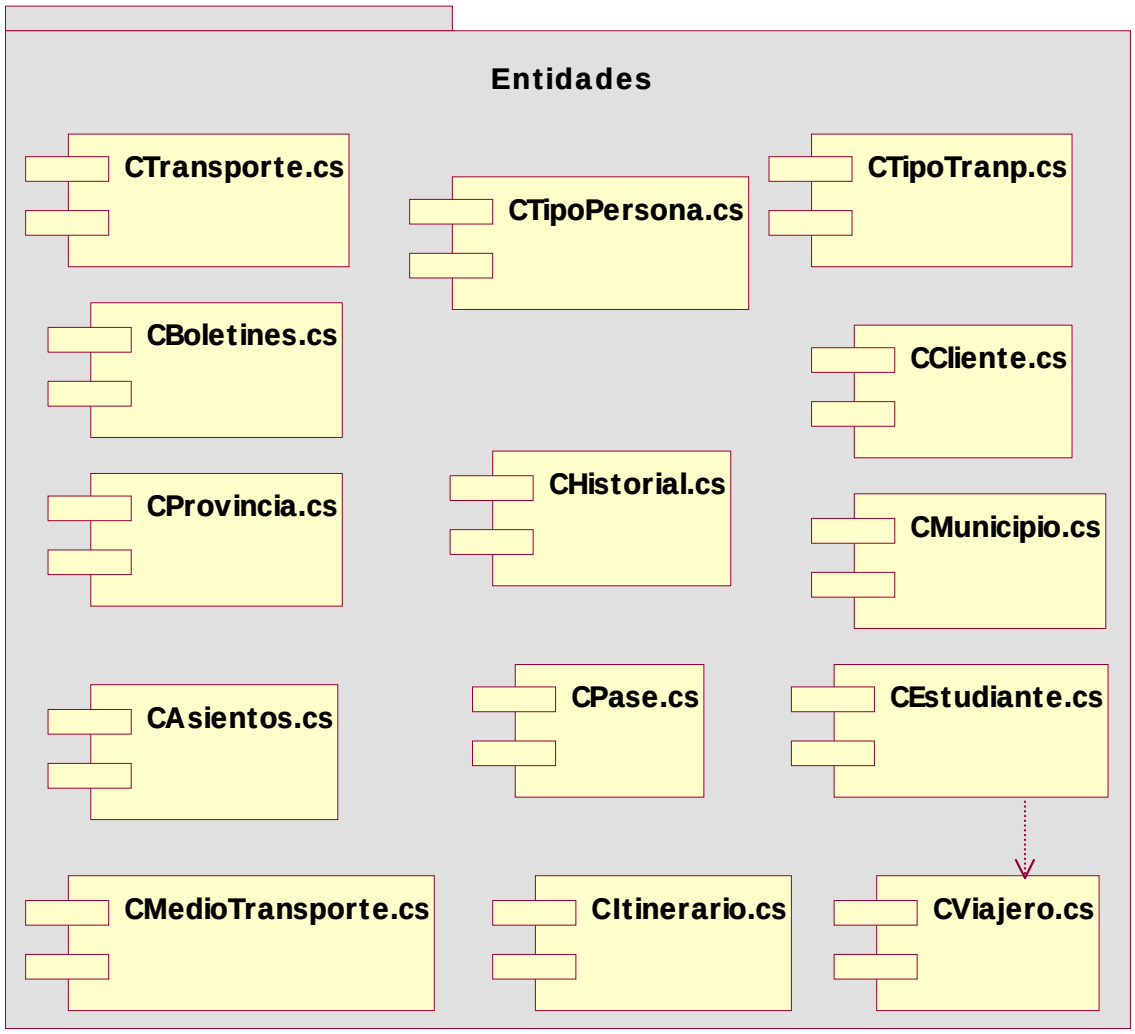


Figura 26. Diagrama de Componentes. Paquete Entidades.

- Paquete Interfaz
- Subpaquete Páginas.

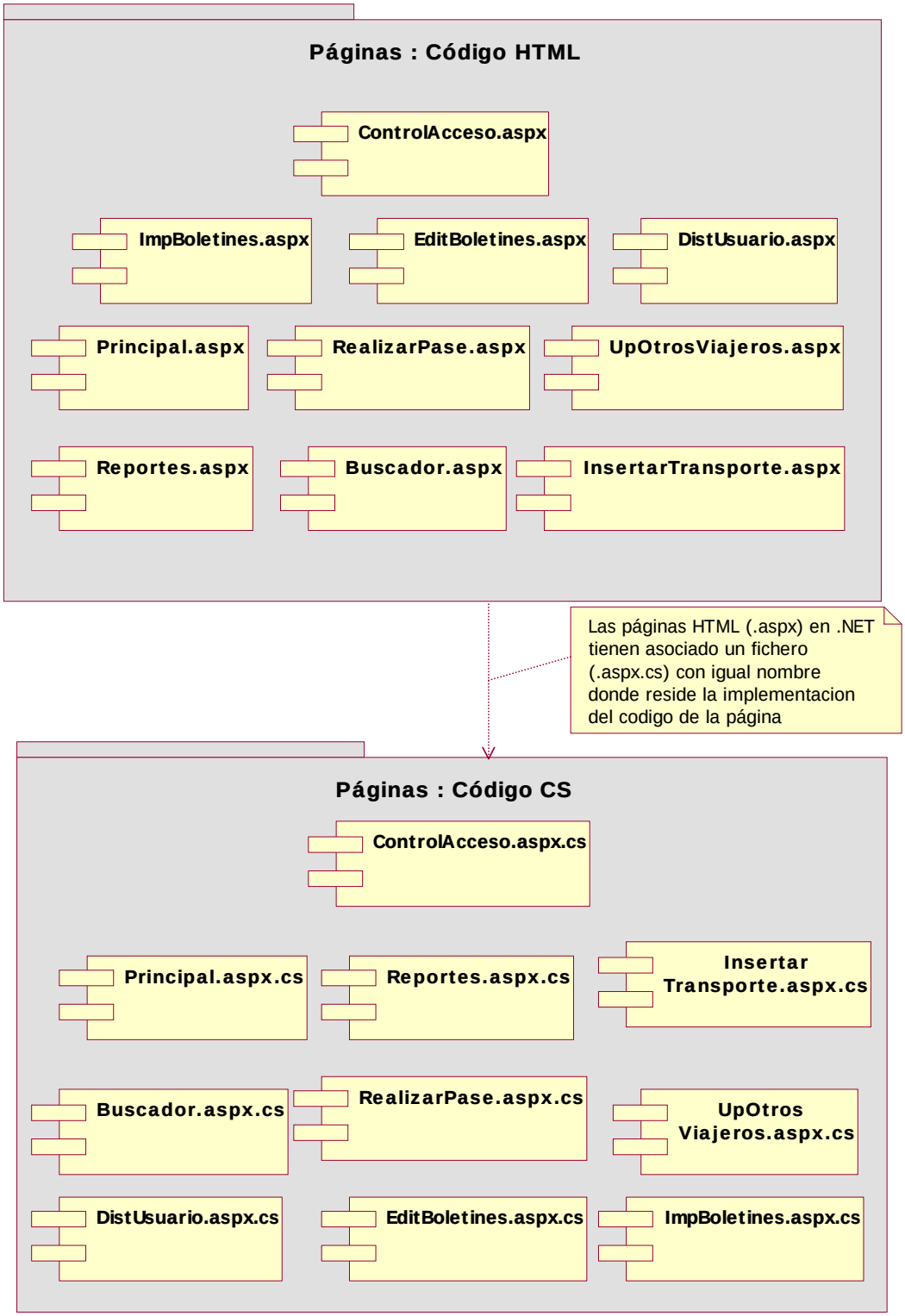


Figura 22. Diagrama de Componentes. Subpaquete Páginas.

- Subpaquete ControlAcceso

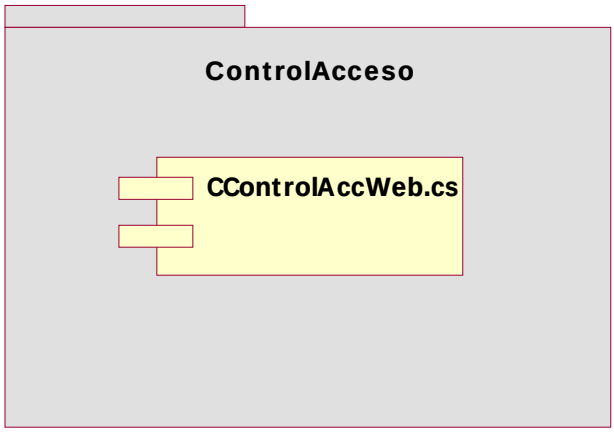


Figura 22. Diagrama de Componentes. Subpaquete ControlAcceso.

4.8 - Conclusiones

Hasta este punto se han descrito las clases y demás elementos necesarios para la implementación del sistema. Se obtuvo el diagrama de clases del sistema y a partir de este se definieron las clases persistentes y se elaboró el modelo de datos. Por otra parte se comentaron los estándares seguidos para el diseño de la aplicación y mediante los modelos de despliegue y de componentes se explicó la estructura física del sistema

.

Capítulo 5

Estudio de factibilidad

5.1 - Introducción

A continuación se hace un estudio de factibilidad y costos del proceso de desarrollo e implementación del Sistema de Reservación de Pase Masivo. Este estudio es de vital importancia antes de empezar el desarrollo de un software pues permite conocer si es conveniente o no trabajar en el mismo. Los beneficios de un proyecto juegan un papel muy importante en la vida del mismo pues la relación entre estos y los costos es lo que permite saber si el proyecto será viable o no. Esta es una relación que se analiza también en el presente capítulo.

5.2 - Planificación

Tabla 1. Entradas Externas

Nombre de la entrada externa	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Adicionar Pase Masivo	1	2	Simple
Adicionar Transporte	2	6	Medio
Adicionar Medio de Transporte	2	7	Medio

Adicionar Historial	1	19	Medio
Adicionar Viajero	1	10	Simple
Adicionar estudiante	2	14	Medio
Eliminar estudiante	2	16	Complejo
Eliminar Pase Masivo	5	16	Complejo
Eliminar Transporte	4	14	Complejo
Eliminar Medio de Transporte	2	8	Medio
Eliminar Viajero	1	12	Simple
Ubicar Viajero	2	2	Simple
Adicionar Familiar	1	1	Simple
Eliminar Familiar	1	1	Simple
Insertar Diseño de Boletines	1	2	Simple
Eliminar Diseño de Boletines	1	2	Simple
Adicionar Cliente	1	1	Simple
Eliminar Cliente	1	1	Simple

Tabla 2. Salidas Externas

Nombre de la salida externa	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Lista de Viajeros	3	7	Medio
Lista de Ubicación	4	13	Complejo
Lista de Estudiantes	5	16	Complejo
Lista de Medios de Transporte	3	4	Simple
Lista de Transportes	4	4	Medio
Historial	1	19	Simple
Lista de Profesores	4	6	Complejo
Lista de Trabajadores	4	6	Complejo
Listado de Familiares	4	6	Complejo
Listado de Dirigentes	4	6	Complejo
Listado de Diseños de Boletines	1	2	Simple

Tabla 3. Peticiones

Nombre de la petición	Cantidad de Ficheros	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
Obtener ubicación de un Viajero	4	13	Complejo
Obtener datos de un	5	7	Complejo

Viajero			
Obtener Familiares de un Profesor	2	7	Medio
Obtener Medios de T. de un Transporte	2	3	Simple
Obtener Transportes de un Pase Masivo	2	3	Simple
Obtener Historial de un Viajero	1	19	Simple

Tabla 4. Ficheros Internos

Nombre del fichero de interfaz interna	Cantidad de records	Cantidad de Elementos de datos	Clasificación (Simple, Media y compleja)
tbl_Asientos	1	5	Simple
tbl_Estudiantes	1	4	Simple
tbl_Historial	1	19	Simple
tbl_Itinerario	1	2	Simple
tbl_MedioTransporte	1	5	Simple
tbl_Municipio	1	4	Simple
tbl_nTipoPersona	1	2	Simple
tbl_nTipoTransporte	1	2	Simple
tbl_Pase	1	3	Simple
tbl_Provincia	1	3	Simple
tbl_Transporte	1	5	Simple

tbl_Viajero	1	12	Simple
tbl_Boletines	1	2	Simple
tbl_Clientes	1	1	Simple

Tabla 5. Puntos de Función desajustados

Elementos	Simples		Medios		Complejos		Subtotal de puntos de función
	No	X Peso	No	X Peso	No	X Peso	
Entradas externas	10	3	5	4	3	6	68
Salidas externas	3	4	2	5	6	7	64
Peticiones	3	3	1	4	2	6	25
Ficheros lógicos internos	14	7	0	10	0	15	98
Total							**255**

5.3 - Costos

Tabla 6. Cantidad de Instrucciones Fuentes

Características	Valor
Puntos de función desajustados	255
Lenguaje	C# 85 %
	SQL 15%
Instrucciones fuentes por puntos de función	59
	39
Instrucciones fuentes por lenguaje (miles de	12.7883

instrucciones)	1.4918
Instrucciones fuentes (miles de instrucciones)	14.28

Tabla 7. Factores de Escala

Factores	Valor	Justificación
PREC	4.96	Es muy diferente a la aplicación que la antecede
FLEX	2.03	Debe cumplir en gran medida con los requerimientos del sistema y de interfaz externa.
RESL	2.83	Se planifican la mayoría de los riesgos críticos
TEAM	0	El equipo de trabajo es altamente cooperativo.
PMAT	6.24	El proceso tiene un alto nivel de madurez (Nivel 1).

Tabla 8. Cálculos Finales de la Estimación de Costos y Esfuerzos

Cálculo de:	Valor	Justificación
Esfuerzo (PM)	42.25	$\Pi EM_i \approx 0.8355$ $E \approx 1.07$ $\sum SF_i = 16.06$ PREC = 4.96 FLEX = 2.03 RESL = 2.83 TEAM = 0.0 PMAT = 6.24
Tiempo de desarrollo (TDEV)	≈11 meses	PM ≈ 42.25 F ≈ 0.31
Cantidad de personas	≈4	PM/TDEV
Costo	≈36 296	C =CHM * PM

	pesos	CHM = 4 * 225 = 859 PM ≈ 42.25
Salario medio	225	El salario mínimo es de 225 pesos.
RCPX	1,33	RELY = alto DOCU = alto CPLX = alto DATA = nominal
RUSE	1,00	Hay una buena concordancia entre la documentación y las necesidades del ciclo de vida.
PDIF	1,00	El .NET es una plataforma estable, no se tienen restricciones considerables de memoria ni de tiempo.
PREX	0.87	La experiencia de los desarrolladores con la plataforma y las herramientas de desarrollo es abundante.
FCIL	0,87	Se usan notablemente las herramientas CASE para documentar e implementar el sistema.
SCED	1,00	Las exigencias en el cumplimiento del cronograma son normales.
PERS	0,83	ACAP = alto PCAP = alto PCON = alto

5.4 - Beneficios tangibles e intangibles

El sistema que se propone no es un sistema para la comercialización, su propósito esta dirigido a resolver las dificultades que se presentan en el proceso de Reservación de Pases Masivos.

Es por ello que el sistema no tiene beneficios tangibles hasta este punto.

Entre los beneficios intangibles tenemos:

- Los Viajeros pueden conocer su Ubicación en los transportes con facilidad y de una forma cómoda y confiable.
- Los trabajadores encargados de hacer la Distribución cuentan con una herramienta segura, eficiente y rápida.
- El tiempo de respuesta ante un Pase Masivo es mucho más corto.
- Los errores se solucionarán de forma sencilla y rápida.
- Los datos estarán protegidos de posible deterioro.
- Los datos estarán disponibles todo el tiempo para su consulta.

5.5 - Análisis de costos y beneficios

El desarrollo e implementación de este sistema no trae consigo grandes gastos de recursos o tiempo. El mismo se nutre de los datos existentes en las bases de datos de la Universidad y puesto que el servidor estará localizado dentro de la misma la conexión es local, lo que posibilita gran rapidez en el acceso a los datos. La tecnología utilizada para el desarrollo del sistema es .NET, que aunque no es gratis, en estos momentos es necesario pagar licencia ya que es un sistema totalmente de uso interno. Es por esta razón fundamentalmente que se ha diseñado el sistema utilizando una estructura multicapas que facilitará el proceso de migración hacia software libre, lo que posibilitará la comercialización de la aplicación.

Todo lo anterior explica por si mismo por qué el sistema no representa un gasto considerable, ya que todo lo que se utiliza para su funcionamiento pertenece a la UCI y sus respuestas son solamente de interés a la UCI.

5.6 - Conclusiones

A lo largo del presente capítulo se ha hecho un estudio de factibilidad y costos del proyecto y se ha concluido que el desarrollo e implementación del mismo no solo es viable sino que aportará grandes beneficios a la Universidad de las Ciencias Informáticas y a todas las personas que en ella residen, trabajan o estudian. Así mismo se pudo conocer los valores de costo de producción, tiempo estimado de realización, esfuerzo que debía realizar el equipo de desarrollo y cantidad de personas necesarias para la realización del sistema, todo esto por medio de la estimación usando COCOMO II basándose en los Puntos de Función desajustados.

Conclusiones.

El presente trabajo se propone brindar una solución eficiente a los problemas encontrados en el proceso de Reservación de Pases Masivos de la UCI. Para ello se sugiere una aplicación capaz de realizar dicha tarea de forma rápida y segura, brindando numerosas variantes y reportes sobre los datos manipulados.

El sistema se ha desarrollado acorde con lo estipulado por RUP y para el modelamiento de cada una de las fases del proyecto se utilizaron las representaciones UML correspondientes.

La solución final provee un ambiente fácil de entender, cómodo, basado en los estándares de diseño y las técnicas más modernas de programación orientada a objetos. Se desarrolló una propuesta de solución y se obtuvieron los casos de uso del sistema y las funciones del mismo partiendo del análisis de los procesos del negocio. Se describieron las clases y elementos necesarios para la implementación del sistema, se obtuvo el diagrama de clases del sistema y a partir de este se definieron las clases persistentes y se elaboró el modelo de datos. Por otra parte se explicaron los estándares seguidos para el diseño del sistema y se logró definir la estructura física del mismo. Posteriormente se analizaron los beneficios y costos en un estudio de factibilidad del proceso que permitió concluir que el sistema no es solamente factible sino que aportará beneficios considerables a la Institución.

A partir de todo esto puede concluirse que el proyecto ha cumplido con los objetivos propuestos de forma satisfactoria. A continuación se refieren algunas recomendaciones para el mejoramiento del sistema que pueden ser de gran utilidad.

Recomendaciones.

A lo largo del desarrollo del presente proyecto, mientras se cumplían los objetivos del mismo, surgieron una serie de ideas para mejorar la efectividad, utilidad y flexibilidad del sistema, las cuales pueden ser implementadas en un futuro y por lo tanto se recomiendan:

- Cambiar la plataforma de desarrollo a software libre, con vistas a la migración que se planea en la UCI.
- Agregar módulos al sistema y mejorar el mismo de manera tal que pueda ser utilizado por cualquier empresa que requiera de una distribución, convirtiéndose en una fuente de ingresos.
- Continuar el desarrollo de este sistema para su correcto acoplamiento al futuro Sistema Integral de Reservaciones de la Universidad de las Ciencias Informática

Bibliografía.

1. Rumbaugh, J.; Jacobson, I. y Booch, G. *The Unified Modeling Language Reference Manual*. Addison-Wesley. 2000.
2. Características de Visual Studio .NET. Microsoft Corp. 2005. http://www.microsoft.com/latam/vstudio/producto/caracteristicas.asp (29/5/2005)
3. Craig, L. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Prentice Hall. 1999.
4. *UML basics: An introduction to the Unified Modeling Language.* http://www128.ibm.com/developerworks/rational/library/769.html (28/5/2005)
5. Pressman, R. *Software Engineering. A Practitioner's Approach*. *Fourth Edition*. McGraw – Hill. 1999.
6. Díaz Ferrandiz, G. *Migrando a ADO.NET.* http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art88.asp (3/6/2005)
7. *¿Qué es ASP.NET?* http://www.zonagratuita.com/microsoft/asp/aspnet.htm (09/06/2005).
8. Schmuller, J. *Aprendiendo UML en 24 horas*, Prentice-Hall, 2001.
9. *¿Qué es .NET?* http://es.wikipedia.org/wiki/.NET (5/6/2005)
10. *Aspectos básicos de XML Wb Services.* http://www.microsoft.com/spanish/msdn/articulos/archivo/280202/voices/webservbasics.asp (5/6/2005)

11. *¿Qué es .NET Framework?* http://www.microsoft.com/latam/net/basics/framework.asp (7/6/2005)

12. *.NET*. Wikipedia. 2005. http://es.wikipedia.org/wiki/.NET_de_Microsoft (1/6/2005)

13. *UML Análisis y Diseño de Software.* http://cfrela.en.eresmas.com/uml/umlanalisis.htm (3/6/2005)

14. *UML.* http://usuarios.lycos.es/oopere/uml.htm (30/5/2005)

15. *C Sharp .NET.* http://es.wikibooks.org/wiki/Programaci%C3%B3n:C_sharp_NET (2/6/2005)

16. Rumbaugh, J.; Jacobson, I. y Booch, G. *El Proceso Unificado del Desarrollo de Software*. Addison Wesley. 2000.

17. Scott, M., Bill, A., Howard, R., Seven, D., Walther, S., Wille, C., Wolthuis D. *ASP.NET: Tips, Tutorials and Code.* Indiana. 2000

18. Francia, J., Henostroza G. *De COM al CLR.* http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art13.asp (1/6/2005)

Glosario de términos

API	Interfaz de Programación de Aplicaciones.
CodeDOM	*Code Document Object Model.* Código del Documento de Modelo de Objetos.
GNU	Licencia que se emite para proteger tanto los derechos de los desarrolladores de software libre como los de los usuarios de este tipo de aplicaciones.
NGWS	*New Generation Windows Services.* Servicios de Windows de nueva generación.
W2K	Siglas que identifican a Microsoft Windows 2000.
W2003	Siglas que identifican a Microsoft Windows 2003.
.NET	Plataforma para desarrollar aplicaciones en casi todos los lenguajes de programación existentes. Creada por Microsoft
Recordset	Objeto capaz de almacenar el resultado de aplicar una sentencia SQL sobre una tabla de la base de datos.