Facultad de Ingeniería y Ciencias Agropecuarias

PORTAL WEB PARA LA FUNDACIÓN COVIAL

Santiago Esteban Flores Ferreira

Sebastián Guillermo Bixby Mera

2011

Facultad de Ingeniería y Ciencias Agropecuarias
Ingeniería en Sistemas

PORTAL WEB PARA LA FUNDACIÓN COVIAL

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero de Sistemas.

Profesor Guía

Ing. Santiago Albuja

Autores

Santiago Esteban Flores Ferreira

Sebastián Guillermo Bixby Mera

2011

Declaración Profesor Guía

Declaro haber dirigido este trabajo a través de reuniones periódicas con los estudiantes, orientando sus conocimientos y competencias para un eficiente desarrollo del tema y tomando en cuenta la Guía de Trabajos de Titulación correspondiente.

Ing. Santiago Albuja
Profesor guía
C.C: 1710245711

Declaración de Autoría del Estudiante

Declaramos que este trabajo es original, de nuestra autoría, que se han citados las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

Santiago Esteban Flores Ferreira
C.C:1715591788

Sebastián Guillermo Bixby Mera
C.C:1713699112

RESUMEN

El objetivo de este proyecto de titulación es la solución a los problemas que los usuarios / clientes de COVIAL tienen que enfrentar para hacer uso de los beneficios que ofrece la fundación, tales como la atención, cuidado, rehabilitación y recuperación de personas que hayan sufrido algún tipo de accidente de tránsito.

Mediante este proyecto se ofrece un servicio ininterrumpido veinte y cuatro horas al día, los siete días de la semana de una manera actual y accesible utilizando un ambiente WEB a través del INTERNET.

Se crea un nuevo programa llamado **WEB COVIAL 1.0**, el que es el encargado de vincular a los usuarios con la fundación, obteniendo información actualizada.

Este nuevo proyecto, tiene además la incorporación de la nueva herramienta Windows Communication Foundation **WCF**, así, de esta manera mantener una conexión mediante Servicios totalmente segura, concurrente y eficiente.

Además, su éxito corresponde al nivel de satisfacción del usuario, por tal razón, WEB COVIAL 1.0 implementa la herramienta de Microsoft, Silverlight, con la finalidad de brindar al usuario una interfaz amigable, intuitiva y de fácil uso, lleno de nuevas virtudes gráficas para mantener el interés constante de los usuarios.

Para ello se detalla la estructura que tiene la fundación, entendiendo sus funciones y objetivos, para luego vincularlos a la obtención de requerimientos de COVIAL.

También se estudian los elementos importantes utilizados para el desarrollo de este trabajo, y de la misma manera, se escoge y estudia la metodología a usar, conocida como RUP.

Se realiza la obtención de Requerimientos de Software siguiendo el estándar *IEEE Std 830-1998*, para lo cual previamente se analiza los requisitos obtenidos inicialmente.

Usando la metodología de Desarrollo de Software RUP, basada en Objetos y UML, se realiza: El diseño, Ingeniería de Negocios, Implementación y Pruebas.

La primera fase, de diseño, abarca un conjunto de diagramas empleados en RUP, que ayudan al entendimiento y visión del programa.

La Ingeniería de Negocios, se refiere a la programación de reglas y clases.

La etapa de Pruebas se realiza identificando y encontrando problemas y errores de programación.

Como producto final se entrega un sitio WEB en el cual el usuario puede interactuar con la fundación al hacer uso de funciones como el chat, foro y de mantenerse al día con noticias que sean relevantes.

ABSTRACT

The purpose of this thesis paper is the solution to the problems that users / customers of COVIAL have to face to use the benefits offered by the foundation, such as attention, care, rehabilitation and recovery of people who have suffered some type of traffic accident.

This project will offer non-stop service twenty-four hours a day, seven days a week making it accessible using a Web environment through the Internet.

It creates a new program called **WEB COVIAL 1.0**, which is responsible for linking users with the foundation, keeping the information up to date. This new project also has the addition of the new Windows Communication Foundation **WCF** tool that allows keeping a connections totally secure, concurrent and efficient.

Its success is the level of user satisfaction, for that reason, COVIAL WEB 1.0 implements the Microsoft tool, Silverlight, in order to give the user a friendly, intuitive and easy to use interface, full of new graphics virtues to maintain the interest of users.
For all this understanding the detailed structure of the foundation, their roles and objectives and then link them to obtain COVIAL requirements.
Important elements for the development of this work are taking care, and in the same way choose and study the methodology, known as RUP.

Software requirements are gathered following the IEEE Std 830-1998 standard, for which the initial requirements are previously discussed.
Using the Software Development methodology known as RUP, based on UML Objects the Design, Business Engineering, Implementation and Testing will be developed.
The first phase is design, which includes a set of diagrams used in RUP, which help the understanding and vision of the program.
Business engineering phase refers to the programming of rules and classes.
The testing phase is done by identifying and finding problems and bugs.

The final product will deliver a website where user can interact with the foundation and make use of features such as chat, forum and keep up with relevant news.

ÍNDICE

1. INTRODUCCIÓN.

Según datos estadísticos las cifras de incidentes de tránsito es alarmante, causa por la cual Ecuador está entre los países con mayor inseguridad vial. En el año 2009 se produjeron más de 13.000 accidentes de tránsito[1].Muchos de los incidentes son causados por falta de conocimiento de las leyes de tránsito y un gran número no son reportados razón por la cual los afectados no tienen asistencia. El funcionamiento de este portal ayudará a Covial a disminuir este gran problema.

1.1. TITULO DEL PROYECTO.

"PORTAL WEB PARA LA FUNDACIÓN COVIAL"

1.2. SITUACIÓN ACTUAL

Para nadie es desconocido que en el Ecuador se vive una inseguridad vial que ha devenido en caos. Accidentes, discapacidad y muerte por doquier. Alarmante porcentaje de daños en la salud física y mental de las personas, muchas veces irreparables. Evidente contaminación ambiental como la polución auditiva y el deterioro paulatino de la calidad del aire de los centros urbanos. Pérdidas materiales y de tiempo, así como retardo en las actividades económicas, que generan millonarias pérdidas al Estado.

Finalmente, nuestra idiosincrasia y atrofiada cultura manifestada en un "que me importa" y anarquía, agravan la crisis.

Dicho de otro modo, la accidentabilidad vial ecuatoriana representa uno de los problemas sociales más graves y acuciantes que se lo soporta con indiferencia, indolencia e irresponsabilidad. Es la causa Nº 1 de mortalidad en niños de 5 a

[1]http://www.eluniverso.com/2010/01/02/1/1447/accidentes-transito-dejaron-muertos-ecuador.html

14 años y la segunda en el grupo de 15 a 49. La pérdida de cinco vidas diarias la sitúan como una de las primeras en mortalidad por factores externos.

Habría unas 85.000 personas en discapacidad por su causa. Y de acuerdo a un estudio en el año 2006 del ex Consejo Nacional de Tránsito y Transporte Terrestres, el aparato productivo ecuatoriano perdió USD$545 millones. En la actualidad esta pérdida sería mayor dado que los accidentes de tránsito aumentan año a año.

La naturaleza de esta cruda realidad es multi-causal y compleja, originada en un problema cultural y de actitud, como así lo sugiere el factor humano causante del 93% de los accidentes de tránsito.

Las soluciones deben darse desde una perspectiva multidisciplinaria y multisectorial, a través de un instrumento único llamado Estrategia Nacional de Seguridad Vial.

1.2.1. CARACTERIZACIÓN DE LA FUNDACIÓN COVIAL.[2]

"Comisión Interinstitucional de Educación, Seguridad y Prevención Vial" (Covial) es una fundación sin fines de lucro. Tiene como objetivo brindar su ayuda a personas quienes han resultado heridas en un accidente de tránsito, además de apoyar y promover a la educación vial en Ecuador.

1.2.1.1. Ubicación

- **Dirección:** Av. Colombia Nº 1227 y Solano, esquina (El Dorado), Casona de la Universidad Internacional del Ecuador, Quito-Ecuador.
- **Telefax:** 02-255-3352
- **Correo electrónico:** covial@covial-ecuador.org

[2] FUENTE: Fundación sin fines de lucro Covial.

1.2.1.2. Misión - Covial.

Covial es una organización de vanguardia, integradora y comprometida, con responsabilidad social y transparencia, que suma esfuerzos y conocimientos humanos, técnicos y científicos.

Fundamentada en valores esenciales: su gente, liderazgo, calidad, solidaridad y mística de servicio; contribuir a construir una cultura de seguridad vial.

1.2.1.3. Visión – Covial.

Ser líder en Educación, Seguridad y Prevención para promover una cultura de seguridad vial que garantice una mejor calidad de vida en nuestro país.

1.2.1.4. Meta – Covial.

Fomentar el mejoramiento permanente de la calidad y estándar de vida en el Ecuador en relación a seguridad vial, coadyuvando a reducir radicalmente el índice de accidentes, a través de planes, programas y proyectos a corto, mediano y largo plazo que, partiendo de la educación y observancia y aplicación de la ley, cubran la totalidad del amplio espectro que presupone la Educación, Seguridad y Prevención Vial en el Ecuador.

1.2.1.5. Objetivo – Covial.

Coadyuvar a convertir al Ecuador en un país educado y seguro en materia vial, siendo la ciudadanía quien tome la iniciativa y liderazgo para propulsar el cambio, a través de un grupo de emprendedores sociales líderes y pioneros, diferentes de la multitud, bienintencionados y creativos, que reuniendo una serie de criterios que juntos caracterizan al emprendedor social excepcional, trabajen en el campo del desarrollo integral de la seguridad vial para salvar vidas.

1.2.1.6. Estructura Organizacional – Covial.

Tabla 1.1: Estructura Organizacional COVIAL 2010.

CARGO	NOMBRE
PRESIDENTE	EC. MARCELO FERNÁNDEZ S.
VICEPRESIDENTE	SR. FRANCISCO BORJA T.
DIRECTOR EJECUTIVO	SR. VÍCTOR JIMÉNEZ E.
SECRETARIO	ING. JORGE BAQUERO A.
TESORERO	ARQ. GALO FLORES G.
PRIMER VOCAL PRINCIPAL	SR. ALEJANDRO VARGAS P.
SEGUNDO VOCAL PRINCIPAL	SR. ERICH GÓMEZ S.
TERCER VOCAL PRINCIPAL	MAGÍSTER MERCEDES SARRADE
CUARTO VOCAL PRINCIPAL	SRTA. MARÍA SUSANA RIVADENEIRA
QUINTO VOCAL PRINCIPAL	SR. ALEX AGUINAGA G.
PRIMER VOCAL SUPLENTE	SRTA. MARÍA LORENA MALDONADO S.
SEGUNDO VOCAL SUPLENTE	SR. FRANK VARGAS P.
TERCER VOCAL SUPLENTE	ING. LUIS FLORES G.
CUARTO VOCAL SUPLENTE	SRA. GABRIELA ZÚÑIGA.
QUINTO VOCAL SUPLENTE	DRA. GUADALUPE VERA.

Fuente: Covial

1.2.1.7. Rol Estratégico.

COVIAL, consciente de que el éxito en la reducción de los accidentes de tránsito y en la obtención de una cultura plena de seguridad vial, provienen del trabajo mancomunado que involucra a los sectores públicos, privados y sociedad civil; lidera iniciativas que aglutinen y potencien el trabajo “en solitario” de muchos, en una sola visión y unión de esfuerzos, para convertir al Ecuador en un país educado y seguro en materia vial.

Sostiene la urgencia de contar con una Estrategia Nacional de Seguridad Vial y viene trabajando arduamente para que se dé pronto. Ejemplo de ello es su "Plan SOS, Paremos la Catástrofe No Natural llamada Accidentes de Tránsito" presentado en enero 2006. El Plan propone un encuentro nacional de emergencia que acuerde medidas coyunturales de choque, alta impacto y aplicación inmediata, que detengan radicalmente el desangre, y sea el punto de partida para crear la estrategia nacional y su plan maestro de acción.

Sostiene que la educación, así como la observancia y aplicación de la ley, son factores clave para convertir al Ecuador en un país culto y seguro. En este contexto, dado que la Educación Vial se sustenta y coadyuva en los fines últimos de la educación, los cuales son formar ciudadanos sanos, cultos, críticos, aptos para la vida y para el ejercicio de la democracia, y capaces de participar en los procesos de transformación social; sostiene que en la estrategia nacional se deben planificar acciones que permitan mejorar cualitativa y cuantitativamente, los modos de actuar de los peatones, conductores y pasajeros.

Mantiene que un trabajo de tal magnitud y complejidad, que sea integral, multidisciplinario, multisectorial, de aplicación inmediata, permanente y sostenible; no puede ser competencia de una, dos o tres entidades, sino de todos.

En tal virtud, por si misma, en mancomunidad de esfuerzos y apoyando y respetando el trabajo y competencias de las instituciones públicas; amparándose en el mandato constitucional de acercar a la sociedad civil e instituciones públicas para incluirlos en procesos de planificación participativa; y, a través de su fuerza que la impulsa, la que se fundamenta en una monolítica identidad de creencias, valores y convicciones que han unificado talento, conocimiento, experiencia, capacidad conductual y mística de servicio a la comunidad; COVIAL trabaja:

- En función de la reducción de los accidentes de tránsito como Política de Estado.
- En apego a la Constitución y Leyes de la República.
- De acuerdo a la información técnica proporcionada por organismos especializados.

Para lo cual tiene un Plan de Acción Marco que consta de cuatro ejes:

- Educar a niños, jóvenes y adultos.
- Asesorar a los gremios.
- Actuar en propuestas específicas y en recepción de denuncias.
- Respaldar a las víctimas, autoridades y líderes de opinión.

El General Francisco Mancajo Gallegos, ex Alcalde del Distrito Metropolitano de Quito, en el Tercer Informe del Observatorio Metropolitano de Seguridad Ciudadana, junio 2004, escribe:*"todo lo que hagamos en pro de mejorar la calidad de vida de nuestros conciudadanos, no tiene precio ni tiempo, es urgente e importante y es tarea de todos"*. Jovial se identifica con esa declaración.

1.3. PROBLEMA A RESOLVER.

1.3.1. DETERMINACIÓN DE PROBLEMAS.

1.3.1.1. Accidentes de Tránsito en Ecuador.

CIFRAS ALARMANTES:

- Ecuador en el 2006 ocupó el 1er lugar en muertes por accidentes de tránsito de Latinoamérica.
- Actualmente está entre los 5 países con mayores accidentes y muertes por tránsito de Latinoamérica.
- 1 de 3 ecuatorianos mayores de 30 años de edad ha sufrido al menos 1 accidente de tránsito en su vida.

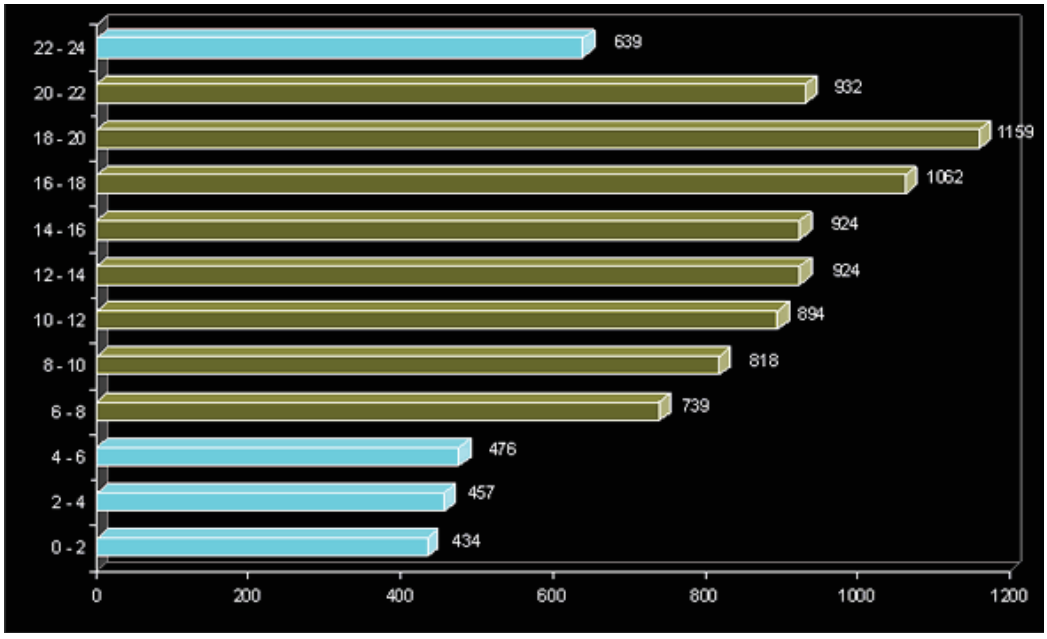
- 7 de 10 ecuatorianos mayores de 30 años tiene algún conocido que ha fallecido en un accidente de tránsito.

DATOS VIALIDAD:

- 980.00 Vehículos.
- 20.000 Accidentes anuales.
- 19.000 Heridos/año.
- 2.530 Muertes/año.
- Quien tiene un accidente de tránsito tiene el 12% de probabilidades de morir.
- 21% de accidentes se da entre las 22h y 6am.

DIAGRAMAS DE ACCIDENTES DE TRÁNSITO EN ECUADOR:[3]

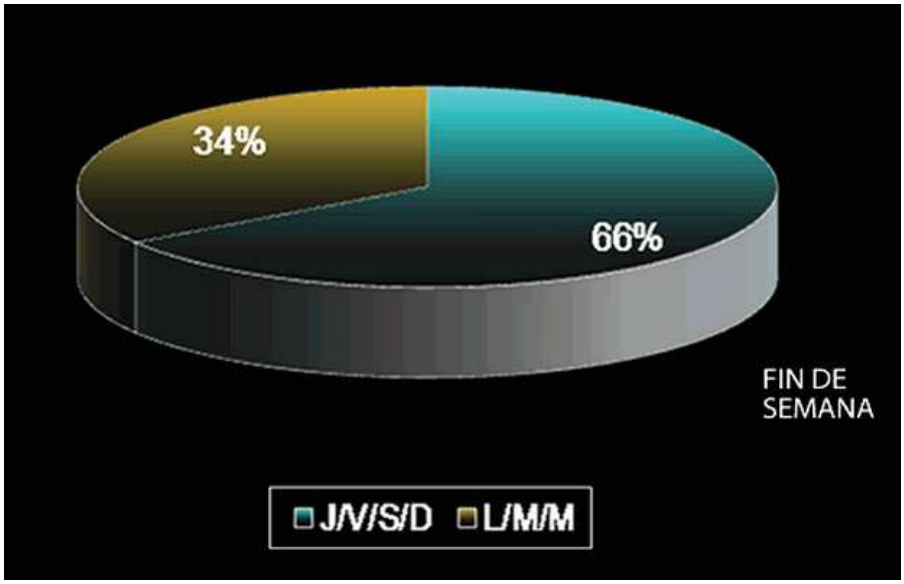
Figura 1.1: Número de Accidentes de tránsito ocurridos por hora. (Enero – Junio 2008).



Fuente: www.conductornominado.com.ec/para-amigos/estadisticas.html

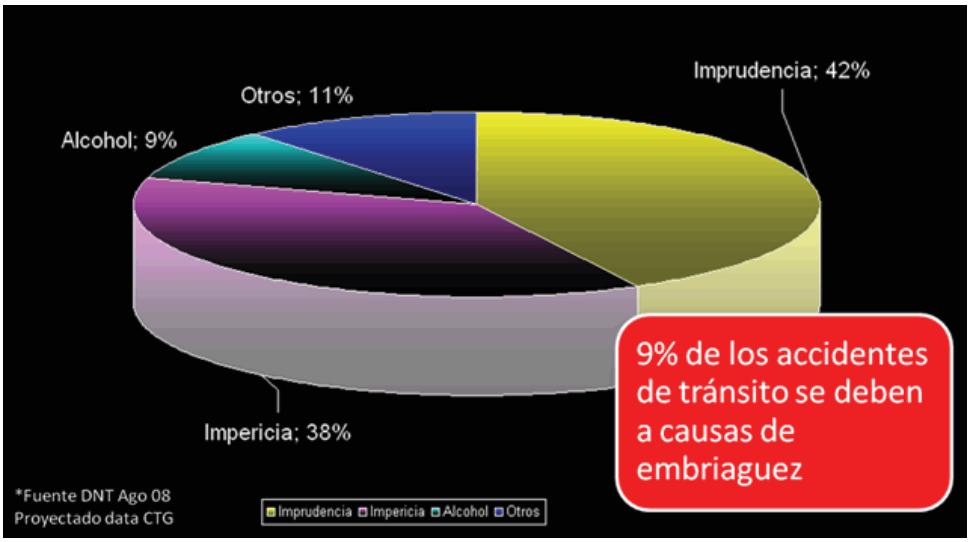
[3] FUENTE : www.conductornominado.com.ec/para-amigos/estadisticas.html

Figura 1.2: Días por accidentes de tránsito



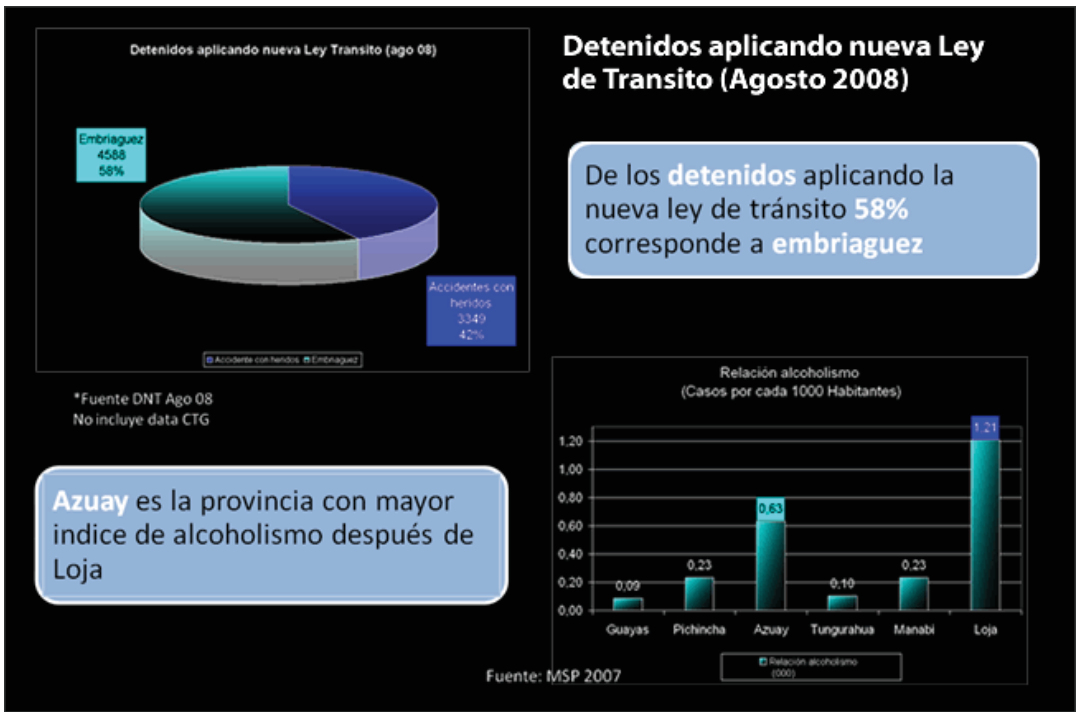
Fuente: www.conductornominado.com.ec/para-amigos/estadisticas.html

Figura 1.3: Causas de Accidentes de Tránsito.



Fuente: www.conductornominado.com.ec/para-amigos/estadisticas.html

Figura 1.4: Detenidos aplicando la Nueva Ley de Tránsito.



Fuente:http://andes.info.ec/ecuador/150-accidentes-de-transito-se-produjeron-durante-el-feriado-en-ecuador-10254.html

Hoy por hoy, existen una gran cantidad de accidentes de tránsito por día, por ejemplo: Durante el feriado de Semana Santa (Abril 2010), la Policía Nacional reportó 150 accidentes de tránsito que provocaron 42 muertes, y 1.561 personas (31 menores de edad) detenidas por diferentes causas. Así informó el director general de Operaciones, Jaime Vaca.[4]

1.4. OBJETIVO GENERAL.

Analizar, diseñar e implementar un sistema de Web para el manejo de la información de incidentes de tránsito y educación vial, para la fundación COVIAL.

[4] FUENTE: "http://andes.info.ec/ecuador/150-accidentes-de-transito-se-produjeron-durante-el-feriado-en-ecuador-10254.html"

1.5. OBJETIVOS ESPECÍFICOS.

- Análisis, diseño e implementación del sistema de información Web utilizando una metodología orientada a objetos.
- Documentar los procedimientos utilizados por la fundación COVIAL para el manejo de información de incidentes de tránsito y educación vial.
- Integrar al sistema de información herramientas y servicios Web que permitan la interacción entre afectados, usuarios y COVIAL a través del Internet.
- Integrar interfaces amigables para el usuario, y que optimicen recursos, mediante el uso de la tecnología multiplataforma Silverlight para aplicaciones Web interactivas.

2. MARCO TEÓRICO

2.1. LEVANTAMIENTO DE INFORMACIÓN

2.1.1. A QUIEN SE DIRIGE EL PROYECTO

Este proyecto de Titulación se encuentra dirigido a los clientes/usuarios de la Fundación COVIAL, para que de esta manera estos se sientan satisfechos con el trato brindado.

El Principal objetivo es brindar al Usuario la manera de conectarse con su fundación haciendo uso de la nueva tecnología, desde cualquier lugar donde exista servicio de Internet; Ofreciendo una interfaz de Usuario amigable, informativa, sencilla y eficiente.

2.1.2. OBTENCIÓN DE REQUERIMIENTOS

En el desarrollo de un Sistema de Software, la obtención de requerimientos es una etapa primordial y crítica; ya que dependiendo de los resultados los usuarios utilizarán o no el sistema, en otras palabras, se determina si el proyecto es factible. Por tanto, se ha creído correcto realizar entrevistas tanto presenciales y escritas, a las personas directamente vinculadas al éxito del mismo.

2.1.2.1. PRINCIPALES PROBLEMAS EN COVIAL.

Si se reacciona ante las cifras detalladas en el Apartado 1.3.1.1, se puede ver la gran cantidad de personas que podrían requerir el servicio de COVIAL.

Sin embargo no todos los individuos que sufren un accidente tienen para sustentar los costos en una clínica u hospital privado aún con la implementación del Seguro Obligatorio de Accidentes de Tránsito (**SOAT**); por otro lado, en los hospitales Públicos, existe una sobrecarga de pacientes, y en muchos de los casos, los pacientes que requieren rehabilitación nunca son

atendidos. Así de la misma manera, COVIAL presenta algunos problemas, de los cuales se habla a continuación.

Ya que el objetivo de este nuevo programa es orientado a la satisfacción del Cliente, se ha recopilado la información mediante encuestas.

Las encuestas se realizaron a Clientes encontrados en COVIAL, en un periodo de 2 meses con los siguientes resultados:

- **Accesibilidad:**
  Actualmente COVIAL enfrenta un gran problema, ya que una persona que desee obtener los beneficios de esta fundación, necesariamente deberá acercarse personalmente a sus instalaciones;
  Por tanto, una gran cantidad de personas que NECESITAN la ayuda de COVIAL no pueden ser ayudados.

- **24 - 7:**
  Hoy por hoy, COVIAL no tiene una apertura para trabajar 24 horas al día y 7 días a la semana, obligando a los clientes a restringirse en sus horas de visita y atención.

  Por otra parte, los clientes, aducen que el sistema telefónico también es ineficiente, ya que, COVIAL al atender gran cantidad de llamadas diarias, hay muchas personas que no logran comunicarse en el momento oportuno con la fundación.

Con la finalidad de tener un alcance total, y realizar un programa completo, las encuestas también fueron realizadas a personal interno de la fundación:

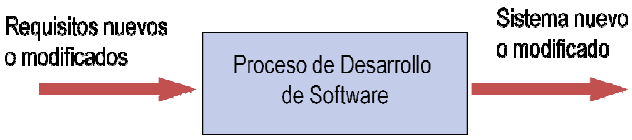
- Empleados internos de la fundación COVIAL, han indicado que muchas veces los Clientes llaman disgustados por la falta de accesibilidad, y de comunicación.

Además citan que ha existido veces que los clientes llegan sin ninguna cita, muchas veces con una emergencia, pero sin embargo no han podido ser atendidos.

2.2. METODOLOGÍA A USARSE.

Una metodología de desarrollo de software es una plantilla para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, definiendo **Quién** debe hacer **Qué, Cuándo** y **Cómo** debe hacerlo.

Figura 2.1: Diagrama de una Metodología



Fuente: http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-en-visual-basic-.net-c273l

El objetivo del proyecto es proporcionar un programa modular y flexible y, para esto, se debe emplear una metodología de desarrollo de software. La metodología que se empleará usa UML (Lenguaje Unificado de Modelado) ORIENTADA A OBJETOS (OO) diseñada por Grady Booch, también conocido como Análisis y Diseño Orientado a Objetos (OOAD).

La metodología OO ha dado un gran avance en la forma de diseño, desarrollo y mantenimiento de software, por lo que se ha decidido tomar esta metodología como referencia por dos razones de peso:

- Brinda una solución al problema que el código no podía ser rehusado ni modificado, o sea la falta de portabilidad de código.

  Las condiciones ya evaluadas pueden cambiar a lo largo del tiempo, por condicionas propias de legislatura o del mercado, por lo que el sistema debe ser realizado necesariamente de manera modular; dado que si en un futuro, COVIAL requiere la incorporación de nuevas funcionalidades o

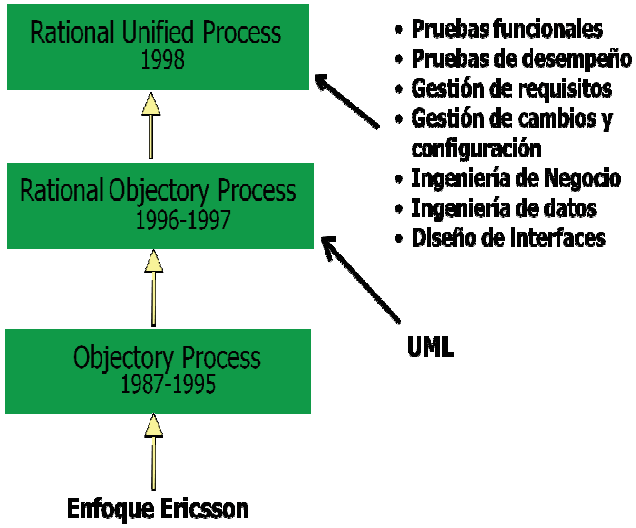
módulos, éstos deben ser incorporados sin necesidad de desechar el programa anterior.

La metodología OO permite realizar un programa totalmente modular y por consiguiente la reutilización y modificación del código.

- La característica principal del proyecto de titulación radica en la utilización de la nueva herramienta WCF (Windows Comunicación Fundación) la cual se encuentra basada en clases (u objetos)[5].

Por tales razones, se ha escogido la metodología **RUP (Rational Unified Procesos)**, la cual no solamente usa UML, sino que a su vez, presenta una guía para todo el proceso de desarrollo (diseño e implementación):

Figura 2.2: Aproximación UML.



Fuente: http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-en-visual-basic-.net-c273l

[5] FUENTE: http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-en-visual-basic-.net-c273l/

Este es un proceso de ingeniería de Software muy bien estructurado, que a la vez permite que la aplicación se adapte a las necesidades y características específicas de la empresa.

Ahora, se definirá esta metodología con más de detalle:

2.2.3. UML (LENGUAJE DE MODELADO UNIFICADO)

"Para la mayoría de las personas, la forma de pensar OO es más natural que las técnicas del análisis y diseño estructurado. Después de todo, el mundo está formado por objetos. Si se piensa un poco, esto tiene sentido. Desde una etapa muy temprana se categoriza los objetos y se descubre su comportamiento. Los usuarios finales y las personas de las empresas piensan de manera natural en términos de objetos, eventos y mecanismos de activación. Se puede crear diagramas OO que les parezcan familiares, mientras tengan dificultades con los diagramas de relación entre entes, tablas de estructura y diagramas de flujo de datos." [6]

A continuación, se presenta un cuadro detallando las principales ventajas al usar una metodología basada en Objetos, y a su vez, sustentado la elección de esta metodología:

Tabla 2.1: Ventajas de una Programación Orientada a Objetos.

FACTOR	DETALLE
Reutilización de Código	El Código mediante objetos y clases permite la reutilización de código.
Estabilidad	Las clases al ser reutilizadas hacen que el programa se comporte de manera más estable.
Complejidad	Términos como encapsulamiento, herencia, polimorfismo hacen que el diseño de algunas clases resulte más sencillo.

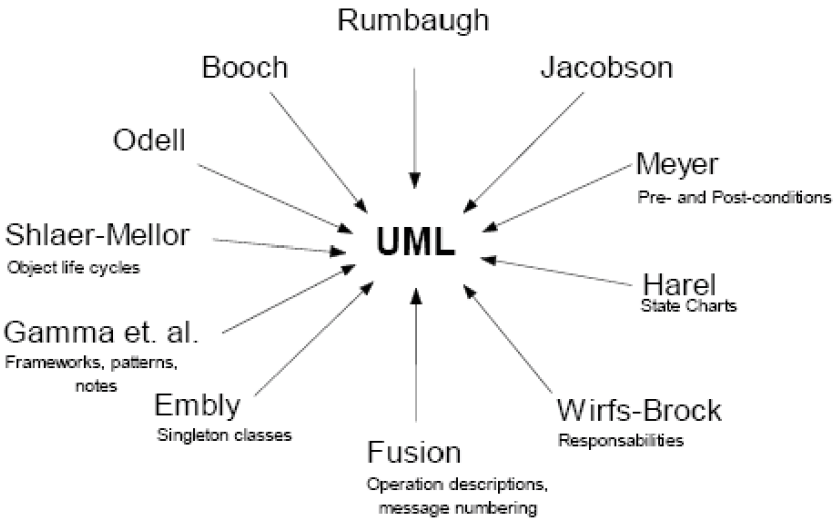
[6]FUENTE: MARTIN- J. J. ODELL

FACTOR	DETALLE
Calidad	Los diseños suelen tener mejor calidad, puesto que se construyen a partir de componentes probados, que han sido verificados y pulidos varias veces.
Sencillez	Los programas se elaboran a partir de piezas pequeñas las cuales ayudan en gran magnitud al desarrollo del mismo.
Mantenimiento	Cada clase efectúa sus funciones independientemente de las demás. Por lo cual los errores serán de fácil ubicación. Los objetos al ser independientes, varios programadores pueden corregir errores al mismo tiempo.
Tiempo	Al poder reutilizar el código es un ahorro significativo de tiempo.
Independencia	Las clases están diseñadas para ser independientes del ambiente, plataformas, hardware y software.

FUENTE: MARTIN- J. J. ODELL

UML es un estándar que nace en 1994, el cual permite especificar, visualizar y construir los artefactos de los sistemas de Software. Actualmente esta metodología agrupa varios enfoques correspondientes a distintos desarrolladores lo que se representa en la siguiente figura:

Figura 2.3: Enfoques en UML.



Fuente: MARTIN- J. J. ODELL

2.2.1.1. Definiciones en UML

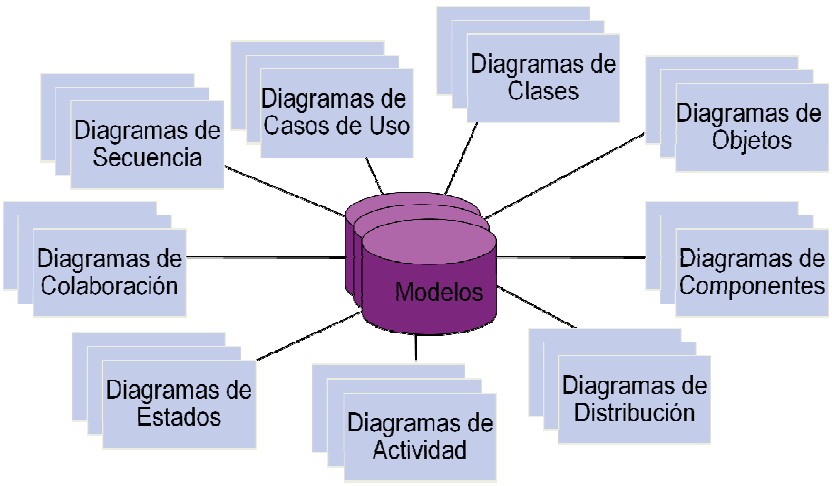
- **Modelo:** captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.
- **Diagrama**: una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un gráfico con vértices conectados por arcos.
- **Paquete**: Ofrece un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Un paquete puede tener sub-paquetes sin límite de anidamiento.
- **Casos de Uso**: Es una técnica para capturar información de cómo el sistema trabaja, o como desea que trabaje.

Estos términos son de importancia al entender que un sistema está compuesto de un conjunto de modelos que permiten expresar a un producto desde cada una de las perspectivas de interés.

El código fuente es el modelo más detallado del sistema, sin embargo es necesario tener otras visiones del sistema.

UML se encuentra conformado de los siguientes diagramas:

Figura 2.4: Diagramas en UML.



Fuente: MARTIN- J. J. ODELL

"Para modelar un sistema es suficiente utilizar una parte de UML, el 80% de la mayoría de los problemas pueden modelarse usando alrededor del 20% de UML**" [Grady Boch].**

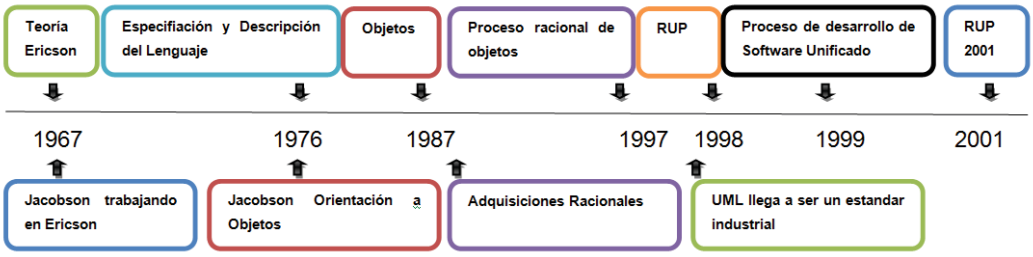
Es por esta razón que no se realizarán todos los diagramas que constan en UML, solamente los más importantes.

2.2.2. RUP

2.2.2.1. Historia de RUP

A continuación, se presenta una figura, la cual detalla la historia que ha venido siguiendo RUP:

Figura 2.5: Historia RUP.



Fuente: www.scielo.org.pe/pdf/iigeo/v10n19/a05v10n19.pdf

El antecedente más importante se ubica en 1967 con la Metodología Ericsson (*Ericsson Aproche*), elaborada por Ovar Jacobson, que es una aproximación de desarrollo basada en componentes que introdujo el concepto de Caso de Uso. Entre los años de 1987 a 1995, Jacobson fundó la compañía *Objetor* AB y lanza el proceso de desarrollo *Objetor* (abreviación de *Objeto Factor*).

Posteriormente en 1995, *Racional Software Corporación* adquiere *Objetor AB* y, entre 1995 y 1997, se desarrolla *Racional Objetor Procesos* (ROP) a partir de *Objetor* 3.8 y del Enfoque Racional (*Racional Aproche*), adoptando UML como lenguaje de modelado.

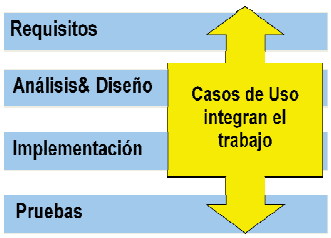
Desde ese entonces y a la cabeza de Grady Boch, Ovar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir ROP, destacándose especialmente el flujo de trabajo conocido como modelado del negocio. En junio del 1998 se lanza *Rational Unified Process*. [7]

[7] FUENTE: http://www.scielo.org.pe/pdf/iigeo/v10n19/a05v10n19.pdf

2.2.2.2. Características Esenciales de RUP

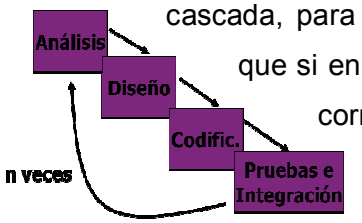
RUP tiene seis características esenciales:

- **Dirigido por Casos de Uso:** La gran ventaja que presenta el usar Casos de Uso es que permiten integrar el trabajo.



Los Casos de Uso son la integración del Trabajo; Hace capturas, validaciones y definición de requisitos, para luego de esto, analizar, diseñar, implementar y probar.

- **Proceso Iterativo e Incremental:** Las actividades se encadenan en una cascada, para luego volverse a repetir con el objetivo de que si en el último escalón se produce un error, se lo corrija siguiendo el mismo proceso.



- **Adaptar el Proceso:** Las características propias del proyecto o empresa (en este caso COVIAL) deben ser totalmente cubiertas por el proceso. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico.

- **Equilibrar Propiedades:** Los requerimientos pueden variar dependiendo de la persona que lo solicite, por tanto, debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio, se podrán corregir desacuerdos que surjan en el futuro.

- **Nivel de Abstracción:** Se deben escoger las herramientas adecuadas, y utilizarlas apropiadamente para que el usuario no perciba nada más que su interfaz gráfica.

- **Calidad:** Debe asegurarse que cada fase lleve un control de Calidad y no esperar al final. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

2.3. CARACTERÍSTICAS DEL SOFTWARE.

2.3.1. DEFINICIONES PREVIAS

2.3.1.1. .NET Framework

Un “Framework” es un soporte definido, en el cual se apoyan otras aplicaciones para poder ser desarrolladas. Generalmente, consta de programas y bibliotecas para ayudar a unir los diferentes componentes de una aplicación.

.NET Framework se encuentra incluido en los Sistemas Operativos: Windows Server 2008, Windows Vista y Windows 7; sin embargo, es posible instalar de forma manual en Windows XP y en la familia de Windows Server 2003. Por otra parte, una versión “reducida” se encuentra en dispositivos inteligentes en el SO Windows Mobile.

Los componentes claves para .NET Framework son:

- **El Entorno Común de Ejecución para Lenguajes o CLR:** Administra el código en tiempo de ejecución, proporcionando los servicios básicos (administración de memoria, control de excepciones, control de hilos de ejecución). Este es el verdadero núcleo de .NET Framework.

  La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio, el MSIL (Microsoft Intermediate Lenguaje), similar al BYTECODE de Java. Para generarlo, el compilador se basa en la especificación CLS (Common Language Specification) que determina las reglas necesarias para crear

el código MSIL compatible con el CLR. Para ejecutarse se necesita un segundo paso, un compilador JIT (Just-In-Time) 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 de hardware. 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 Biblioteca de Clases Base o BCL de .NET Framework:** Define un conjunto funcional mínimo para que sea soportado por un Sistema Operativo. Contiene los tipos básicos como clases de entrada/salida de datos (XML, TCP/IP), seguridad, manejo de dispositivos periféricos, etc.

  Incluye dos componentes ADO.NET y ASP.NET:

  - **ADO.NET** se deriva de ActiveX Data Object y es usado para acceso a información y servicios de datos en un SGBD (Sistema de gestor de Base de Datos).
  - **ASP.NET** no es otra cosa que un Framework para aplicaciones web. Su primera aparición fue junto con la versión de .NET Framework 1.0.

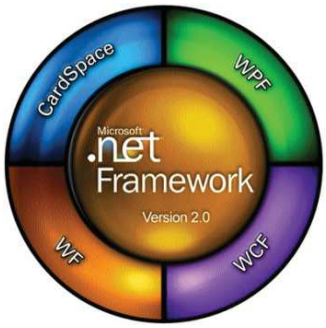
- **El conjunto de lenguajes de programación:** Gracias al CLI (Infraestructura común de Lenguajes)[8], actualmente .NET soporta ya más de cincuenta lenguajes de programación, dando una gran versatilidad en el desarrollo de un programa, sin tener mayores diferencias si se programa con uno u otro lenguaje.

[8]CLI es una especificación estandarizada cuya característica esencial es permitir que diferentes lenguajes de alto nivel puedan ejecutarse en múltiples plataformas sin necesidad de recompilar el código fuente.

Entre los lenguajes más usados en el desarrollo .NET están: C#, Visual Basic .NET, Delphi (Object Pascal), C++, J#, Perl, Python, Fortran, Cobol y PowerBuilder.

WPF para poder tener todo su potencial en marcha sugiere un Framework versión 3.0 o superior, sin embargo Microsoft ha lanzado el primer Service Pack para su Framework .NET 2.0. Entre las mejoras que proporciona este Service Pack se puede destacar mejoras de seguridad importantes y soporte a los prerrequisitos de .NET Framework 3.0 Service Pack 1 y .NET Framework 3.5.

Figura 2.6: WPF y .Net Framework 2.0.[9].



Fuente: www.visualbeta.es/wp-content/uploads/2007/12/Framework-net-2_0.jpg

2.3.1.2. Kit de Desarrollo de Software de Windows (SDK)

SDK se presenta como un API incluyendo herramientas, documentación y ejemplos que necesitan los desarrolladores para escribir, generar, probar e implementar aplicaciones de .NET Framework. Es compatible con los SO de las familias Windows 2000; Windows Server 2003; Windows Vista, Windows XP, Windows 7.

Los SDKs frecuentemente incluyen, también, códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario.

[9]FUENTE: http://www.visualbeta.es/wp-content/uploads/2007/12/Framework-net-2_0.jpg

2.3.1.3. Entorno Integrado de Desarrollo (IDE)

Un IDE es un entorno de programación que suele consistir de un editor de código, un compilador y un constructor de interfaz gráfica.

Visual Studio 2008 es un IDE creado por Microsoft®. Éste presenta un conjunto de herramientas destinadas a escribir y modificar código, además a detectar y corregir errores en las diferentes aplicaciones. Soporta varios lenguajes de programación y se han desarrollado las extensiones necesarias para muchos otros no nativos de Microsoft®.

Figura 2.7: Visual Studio 2008 (Logo).

Fuente: msdn.microsoft.com/es-es/vstudio/2010.aspx

El nuevo Framework (.Net 3.5) con el que trabaja Visual Studio 2008 está diseñado para aprovechar las ventajas que ofrece Windows Vista y el actual Windows 7 a través de sus subsistemas **"Windows Communication Foundation" (WCF)** y "Windows Presentation Foundation" (WPF).

Es importante destacar que si se instala Microsoft Visual Studio 2005 o posterior (excepto las ediciones Express), no es necesario instalar el SDK de .NET Framework por separado.

Visual Studio 2010 crea aplicaciones enriquecidas para SharePoint y la Web[10] e incorpora nuevas características mejoradas que hacen que todo el proceso

[10]**Fuente:** http://msdn.microsoft.com/es-es/vstudio/2010.aspx

de desarrollo sea más sencillo, desde el diseño a la implementación. Personaliza el área de trabajo mediante la compatibilidad con varios monitores.

Los requisitos[11] que debe tener un sistema para un correcto funcionamiento en este IDE son:

- **Sistema Operativo:** Windows Server 2003; Windows Server 2008; Windows Vista; Windows XP, y actualmente Windows 7.
- **Mínimo:** 1.6 GHz CPU, 384 MB RAM, pantalla 1024x768, disco duro 5400 RPM
- **Recomendado:** 2.2 GHz o mayor de CPU, 1024 MB o mayor en RAM, pantalla de 1280x1024, 7200 RPM o mayor de Disco Duro.
- **Sobre Windows Vista o Windows 7:** 2.4 GHz CPU, 768 MB RAM.

2.3.1.4. Interfaz de Programación de Aplicaciones (API)

La API es un biblioteca, generalmente dinámica o DLL (Dinamic Link Library), utilizada para conseguir abstracción en la programación de las capas inferiores, brindando simplicidad en la programación, o incluso independencia del SO o del lenguaje de programación.

Visto desde código en bajo nivel (lenguaje de máquina), tiene como propósito proporcionar un grupo de funciones y procedimientos de uso general. La gran ventaja radica en que el programador evita el trabajo de programar funcionalidades ya existentes. Sin embargo, las APIs de alto nivel generalmente pierden flexibilidad; por tanto, al elegir una API se debe llegar a un equilibrio entre su potencia, simplicidad y pérdida de flexibilidad.

[11] **Fuente:** http://www.microsoft.com/downloads/details.aspx?FamilyID=75cbcbcd-b0e8-40ea-adae-85714e8984e3&displaylang=en#Requirements

Entre las APIs más destacadas se encuentran: Microsoft WMI, Microsoft Win32 API, Microsoft Framework .NET, OpenGL, Java EE, entre otros.

2.3.1.5. Kit de Desarrollo SDK

SDK se presenta como un API incluyendo herramientas, documentación y ejemplos que necesitan los desarrolladores para escribir, generar, probar e implementar aplicaciones de .NET Framework. Es compatible con los SO de las familias Windows 2000; Windows Server 2003; Windows Vista, Windows XP, Windows 7.

Los SDKs frecuentemente incluyen, también, códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario.

2.3.1.6. Lenguaje de Marcado (XAML)

Es más conocido mediante las siglas XAML, y obviamente su estructura es basada en XML. Es utilizado para crear la GUI, con herramientas comunes como ventanas, cuadros de diálogo, controles de Usuario, gráficos, etc.

Etiquetas o tags descriptivas dentro de caracteres < y >. Cada elemento es iniciado con un nombre específico y siempre se debe señalar su cierre utilizando el mismo nombre más los caracteres </, lo que especificará la finalización del tag. En la siguiente figura, también se puede notar cómo un elemento puede contener otros elementos, así como atributos descriptivos y valores dentro del mismo nodo. XAML sigue este mismo patrón:

Figura 2.8: Lenguaje XAML.

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Window with Button"
    Width="250" Height="100">

  <!-- Add button to window -->
  <Button Name="button">Click Me!</Button>

</Window>
```

Fuente: msdn.microsoft.com/es-es/library/cc189054(vs.95).aspx

XAML es uno de los componentes fundamentales de Silverlight. XAML es un lenguaje declarativo, formado por etiquetas descriptivas para cada uno de los componentes que Silverlight pueda representar. XAML basa el concepto descriptivo en el conocido modelo de XML. Gracias a esta cualidad (base de XML), entender la estructura de XAML resulta simple e intuitivo.

VER ANEXO 1

2.3.2. ANÁLISIS DE LAS HERRAMIENTAS DE SOFTWARE

2.3.2.1. Windows Comunication Foundation (WCF)

Desarrollado por Microsoft, es un modelo de programación unificado que permite a los programadores generar soluciones con transacción seguras y de confianza, que se integren en diferentes plataformas y que inter-operen con las inversiones existentes.

Debido a que el mecanismo de comunicación fundamental de WCF es un servicio Web basado en SOAP, las aplicaciones basadas en WCF pueden comunicarse con otro software que se ejecute en una variedad de contextos. Una aplicación generada en WCF puede interactuar con todo lo siguiente:

- Las aplicaciones basadas en WCF que se ejecutan en un proceso diferente en el mismo equipo de Windows.
- Las aplicaciones basadas en WCF que se ejecutan en otro equipo de Windows.
- Las aplicaciones generadas en otras tecnologías, como servidores de aplicaciones de J2EE, que son compatibles con los servicios Web estándar. Estas aplicaciones se pueden estar ejecutando en equipos con Windows o en los equipos que ejecutan otros sistemas operativos.

Mensajería y extremos[12]

WCF se basa en la noción de comunicación basada en mensajes y cualquier cosa que se pueda modelar como un mensaje (por ejemplo, una solicitud HTTP o un mensaje de MSMQ), se puede representar de manera uniforme en el modelo de programación. Esto habilita una API unificada en todos los mecanismos de transporte diferentes.

El modelo distingue entre *clientes*, que son aplicaciones que inician la comunicación y *servicios*, que son aplicaciones que esperan a que los clientes se comuniquen con ellos y respondan a esa comunicación. Una única aplicación puede actuar como cliente y como servicio.

Los mensajes se envían entre extremos. Estos son los lugares donde los mensajes se envían o reciben (o ambos), y definen toda la información requerida para dicho intercambio. Un servicio expone uno o más extremos de la aplicación (y a cero o más extremos de la infraestructura), y el cliente genera un objeto que sea compatible con uno de los extremos del servicio.

Un *extremo* describe de una manera basada en estándar dónde se deberían enviar los mensajes, cómo se deberían enviar y qué aspecto deberían tener los mensajes. Un servicio puede exponer esta información como metadatos que

[12] FUENTE: http://msdn.microsoft.com/es-es/library/ms731079(v=VS.90).aspx

los clientes pueden procesar para generar clientes WCF adecuados y pilas de comunicación.

Protocolos de comunicaciones

Un elemento requerido de la pila de la comunicación es el *protocolo de transporte*. Los mensajes se pueden enviar a través de intranets e Internet utilizando transportes comunes, como HTTP y TCP. Otros transportes incluidos admiten la comunicación con aplicaciones Message Queue Server de Microsoft (MSMQ) y nodos en una malla de redes del mismo nivel. Se pueden agregar más mecanismos de transporte utilizando los puntos de la extensión integrados de WCF.

Otro elemento necesario en la pila de comunicación es la codificación que especifica cómo se da formato a cualquier mensaje determinado. WCF proporciona las siguientes codificaciones:

- Codificación de texto, una codificación interoperable.
- Codificación Mecanismo de optimización de transmisión de mensajes (MTOM), que es una manera interoperable de enviar eficazmente datos binarios no estructurados a y desde un servicio.
- Codificación binaria para una transferencia eficaz.

Se pueden agregar más mecanismos de codificación (por ejemplo, una codificación de compresión) utilizando los puntos de extensión integrados de WCF.

Patrones de mensajes

WCF admite varios patrones de mensajería, incluida la comunicación de solicitud-respuesta unidireccional y dúplex. Los transportes diferentes admiten patrones de mensajería diferentes y, por consiguiente, afectan a los tipos de interacciones que admiten. El tiempo de ejecución y las API de WCF también le ayudan a enviar mensajes de manera segura y fiable.

Requisitos WCF

WCF está instalado de manera predeterminada en Windows Vista y Windows 7.

WCF también se pueden instalar en Windows XP SP2, Windows Server 2003 R2 o Windows Server 2003 SP1.

Nota: La funcionalidad de Message Queue Server (MSMQ) de WCF sólo se admite en Windows Vista, Windows Server 2003 R2, Windows Server 2003 SP1 y Windows XP Professional.[13]

Para permitir más que únicamente la comunicación básica, WCF implementa tecnologías de servicios Web definidas por las especificaciones de WS-*. Todas estas especificaciones fueron definidas originalmente por Microsoft, IBM y otros proveedores que trabajan juntos. Cuando las especificaciones se estabilizan, la propiedad pasa a menudo a los organismos de creación de estándares, como el World Wide Web Consortium (W3C) u OASIS (Organization for the Advanced of Structures Information Standards, Organización para el avance de estándares de información estructurada). Estas especificaciones tratan varias áreas, incluidas la mensajería básica, la seguridad, la confiabilidad, las transacciones y el trabajo con los metadatos de un servicio.

A continuación se detallan 4 especificaciones para WCF:

Tabla 2.2: Especificación WCF.

Especificación	Detalle
Mensajería: SOAP	SOAP es la base para los servicios Web y define un sobre básico que contiene las secciones de encabezado y de cuerpo. WS-Addressing define las sumas al

[13]**Fuente**: http://msdn.microsoft.com/es-es/library/ms733890.aspx

Especificación	Detalle
	encabezado SOAP para direccionar mensajes SOAP, que evita que SOAP confíe en el protocolo de transporte subyacente, como HTTP, para realizar el direccionamiento de la información. El mecanismo de optimización de transmisión del mensaje (MTOM) define un formato de transmisión optimizado para los mensajes SOAP con mucho contenido de datos binarios.
Metadatos	El lenguaje de descripción de servicios web (WSDL) define un lenguaje estándar para especificar servicios y varios aspectos sobre cómo se pueden utilizar esos servicios. **WS-Policy** permite la especificación de aspectos más dinámicos del comportamiento de un servicio que no se pueden expresar en WSDL, como una opción de seguridad preferida. WS-MetadataExchange permite que un cliente solicite directamente información descriptiva sobre un servicio, como su WSDL y sus directivas, utilizando SOAP.
Seguridad	WS-Security, WS-SecureConversation, WS-Trust y WS-Federation todos definen las sumas a los mensajes SOAP para proporcionar autenticación, integridad de datos, privacidad de datos y otras características de seguridad.
Confiabilidad	La mensajería WS-Reliable define las sumas al encabezado SOAP que permiten la comunicación confiable de un extremo a otro, incluso cuando se deben atravesar uno o más intermediarios de los servicios Web.
Transacciones	Se genera en WS-Coordination, la transacción WS-Atomic permite coordinar las transacciones de la

Especificación	Detalle
	confirmación en dos fases en el contexto de conversaciones de servicios Web.

Fuente: http://msdn.microsoft.com/es-es/library/ms733890.aspx

Seguridades en WCF

Tabla 2.3: Seguridades en WCF.

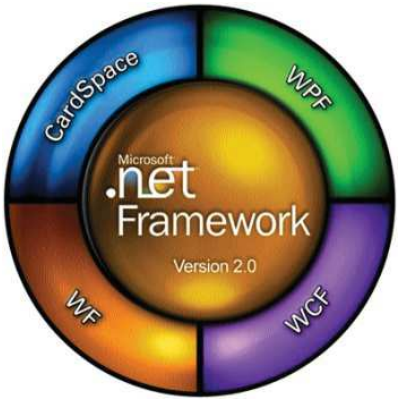
Parámetros	Descripción
Seguridad global	Un transporte seguro, como Capa de sockets seguros (SSL) solo funciona cuando la comunicación es de punto a punto. Si el mensaje se dirige a uno o más intermediarios de SOAP antes de alcanzar el receptor final, el propio mensaje no está protegido cuando un intermediario lo lee desde la conexión. Además, la información de autenticación del cliente está disponible solamente para el primer intermediario y debe ser transmitida al último recibido en modo fuera de banda, si es necesario. Esto se aplica incluso en el caso de que la ruta completa utilice la seguridad de SSL entre los saltos individuales.
Flexibilidad	Se pueden firmar o cifrar las partes del mensaje, en lugar del mensaje completo. Esto significa que los intermediarios pueden ver las partes del mensaje diseñadas para ellos. Si el remitente necesita que parte de la información en el mensaje sea visible a los intermediarios, pero desea asegurarse que no se manipula, simplemente puede firmarlo sin cifrarlo. Puesto que la firma forma parte del mensaje, el receptor final puede comprobar que ha recibido la información del mensaje intacta. Un escenario podría tener un servicio intermediario de SOAP que dirige el mensaje según lo especificado en el valor de encabezado de Acción. De forma predeterminada, WCF no

Parámetros	Descripción
	cifra el valor de la acción sino que lo firma si se utiliza la seguridad de mensajes. Por consiguiente, todos los intermediarios pueden tener acceso a esta información, pero nadie puede cambiarla.
Compatibilidad (Transporte)	Puede enviar mensajes seguros sobre muchos transportes diferentes, como canalizaciones con nombre y TCP, sin tener que confiar en el protocolo para la seguridad. Con seguridad de nivel de transporte, toda la información de seguridad es delimitada a una conexión de transporte determinada única y no está disponible desde el propio contenido del mensaje. La seguridad de mensaje protege el mismo sin tener en cuenta el transporte que se utiliza para transmitirlo y el contexto de seguridad se incrusta directamente.
Compatibilidad (Credenciales y Notificaciones)	La seguridad del mensaje está basada en la especificación WS-Security, que proporciona un marco extensible capaz de transmitir cualquier tipo de notificación dentro del mensaje SOAP. A diferencia de la seguridad de transporte, el conjunto de mecanismos de autenticación, o las notificaciones, que puede utilizar no están limitados por las funciones del transporte. La seguridad del mensaje WCF incluye varios tipos de autenticación y transmisión de notificación y se puede extender para admitir tipos adicionales, si es necesario.

Fuente: http://msdn.microsoft.com/es-es/library/ms733890.aspx

2.3.2.2. Silverlight 2.0

Figura 2.9: Framework Versión 2.0.



Fuente:www.silverlight.net

Silverlight 2 usa, para la representación visual de sus elementos, XAML. Las mismas etiquetas que también son soportadas por aplicaciones de escritorio construidas con WPF.

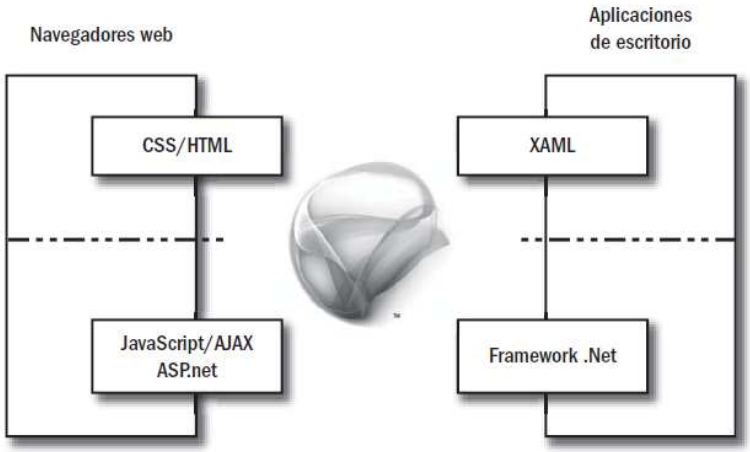
En la Web, las aplicaciones tienen más representatividad y, por ende, su atractivo y su facilidad de uso necesitan potenciarse. Esta mejora es llevada a cabo por la implementación de soluciones que conjugan JavaScript y XML, dando como producto la pieza conocida como A.J.A.X., donde bien se puede encontrar cientos de modelos listos para ser implementados.

Silverlight se nutre de cada una de las principales características de los dos mundos, generando una mejor experiencia de usuario e independencia en plataformas.

Por un lado, los controles y componentes versátiles propuestos por el desarrollo de escritorio y, por el otro, la portabilidad entre plataformas y la rapidez de la actualización de aplicaciones a miles de usuarios al mismo tiempo que otorga Internet.

Esta versatilidad y portabilidad entre plataformas, tanto de escritorio como web, no sería posible sin un lenguaje común de representación de controles.

Figura 2.10: Diseño de Silverlight.



Fuente:www.silverlight.net

XAML es el lenguaje que dará esta posibilidad no sólo para el desarrollo, sino que también brinda la oportunidad de aumentar la versatilidad de las interfaces, poder llevar el comportamiento de las aplicaciones de escritorio a la Web, posibilitar la adopción por parte del usuario final de manera independiente del sistema, mediante el aumento de la calidad y de la velocidad en el desarrollo, y otros puntos ganados con la utilización de XAML y Silverlight.

Silverlight también presenta una gran cantidad de funcionalidad en cuanto a su presentación visual y puede manejar sonidos, imágenes y videos, así como una serie de controles y componentes listos para usar.

Tabla 2.4: Funcionalidades Silverlight.

Componente	Descripción / Detalle
Video y sonido	Inclusión de soporte de formatos de video y sonido comunes como MP3 y WMA. Incluye capacidades de streaming
Imágenes	Capacidad de despliegue de imágenes tanto vectoriales como mapas de bits en sus formatos más comunes, texto

Componente	Descripción / Detalle
	y animaciones.
Enlazado de datos	Capacidad de enlazado de fuentes de datos automática que facilita el despliegue de la información desde diferentes fuentes de datos
Controles	Conjunto de controles listos para usar que brindan la posibilidad de crear nuestro propio set de controles y de rehusarlos en diferentes aplicaciones
XAML	Implementación de Extensible Application Markup Language para la confección de las interfaces. Éstas son creadas sobre la base de XML.

Fuente:www.silverlight.net

Microsoft provee de dos herramientas principales, cada una de ellas enfocada a cubrir una de las ramas involucradas en la generación de elementos Silverlight. Por un lado, para desarrolladores de software, la herramienta por excelencia es Microsoft Visual Studio en su versión 2008.

Aunque la nueva versión de esta herramienta puede obtenerse con facilidad (Visual Studio 10), la que se ha nombrado es suficiente para emprender nuestra labor. Microsoft Visual Studio 2008 resultará ideal debido a su soporte de Microsoft .Net Framework 3.5, herramienta que trae consigo una serie de plantillas para Silverlight 2.

Herramientas de Desarrollo Silverlight:

Una de las principales cualidades de Silverlight es el potenciamiento visual en interfaces de usuario. Así, de manera sencilla, ayudará en la confección de los elementos XAML, su estructura, controles anidados, enlazado a fuentes de datos, entre otros.

A continuación se detalla las principales características por las que se ha usado esta herramienta en este proyecto:

Tabla 2.5: Características de Herramientas usadas.

Característica	Descripción
WCF, WPF, XAML	Silverlight incluye un subconjunto de la tecnología Windows Presentation Foundation (WPF), que extiende en gran medida los elementos en el explorador para crear la interfaz de usuario. Silverlight permite crear gráficos, animaciones y elementos multimedia fascinantes, así como otras características de cliente enriquecidas, extendiendo la interfaz de usuario basada en explorador más allá de lo que está disponible únicamente con HTML. El Lenguaje XAML proporciona una sintaxis de marcado declarativa para crear elementos.
Extensiones a JavaScript	Silverlight proporciona extensiones al lenguaje de scripting de explorador universal que permiten controlar la interfaz de usuario del explorador, incluida la capacidad para trabajar con elementos WPF.
Compatibilidad con varios exploradores y plataformas	Silverlight se ejecuta de la misma manera en todos los exploradores conocidos (y en las plataformas conocidas). Es posible diseñar y desarrollar aplicaciones sin tener que preocuparse del explorador o de la plataforma de los usuarios.
Integración con aplicaciones existentes	Silverlight se integra perfectamente con el código JavaScript y ASP.NET AJAX existente de modo que complementa la funcionalidad ya creada.
Acceso al modelo de programación de .NET Framework	Es posible crear aplicaciones de Silverlight mediante lenguajes dinámicos, como IronPython, y lenguajes como C# y Visual Basic
Compatibilidad de herramientas	Se pueden utilizar herramientas de desarrollo, como Visual Studio y Expression Blend, para crear rápidamente

Característica	Descripción
	aplicaciones de Silverlight.
Compatibilidad de red.	Silverlight incluye compatibilidad con HTTP sobre TCP. Se puede conectar a los servicios WCF, SOAP o ASP.NET AJAX y recibir datos XML, JSON o RSS. Además, es posible compilar clientes de multidifusión con Silverlight
LINQ	Silverlight incluye Language Integrated Query (LINQ), que permite programar el acceso a datos utilizando una sintaxis nativa intuitiva y objetos fuertemente ligados en los lenguajes de .NET Framework

Fuente:www.silverlight.net

2.3.2.3. Modelo Vista Controlador

Este es un patrón de diseño de arquitectura asociado a la idea de 3 capas; sirve para brindar una nueva estructura lógica a las aplicaciones, donde su principal objetivo es la separación de la lógica de negocio de la interfaz y los datos o información, para obtener reusabilidad de código.

Este patrón ya tiene mucho tiempo trabajando, su primera aparición fue en 1979 por TrygveReenskaug, quien trabajó en Smalltalk en un laboratorio de investigación de Xerox.[14]

Los participantes en este patrón son:

- *Usuario o Actor:*

  Es la persona quien manipula el programa, o sea puede acceder a la Vista y al Controlador para manipular el Modelo.

- *Modelo: Los datos junto con la lógica del modelo*

[14]**Fuente**:http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Tiene dos tareas básicamente, la primera es que debe tener la facilidad de acceder al almacenamiento de datos, el cual puede ser una base de datos, y por otro lado, su segunda tarea es definir las reglas del negocio o la funcionalidad del sistema.

- *Vista: Representación gráfica del modelo.*

Se lo llama de esta manera puesto que es la apariencia que tomará la aplicación al interactuar con el usuario, entre sus tareas más relevantes se citan:

- Recopilar la información importante para luego ser mostrada al Usuario mediante su interfaz.
- Debe tener un registro controlador, ya que, este será su aliado para poder realizar lo que el usuario desee.
- Actualiza los datos, para que estos siempre se encuentren disponibles para el Usuario o Actor.

Una vista puede estar asociada a un modelo, y también varias vistas pueden estar asociadas a un mismo modelo.

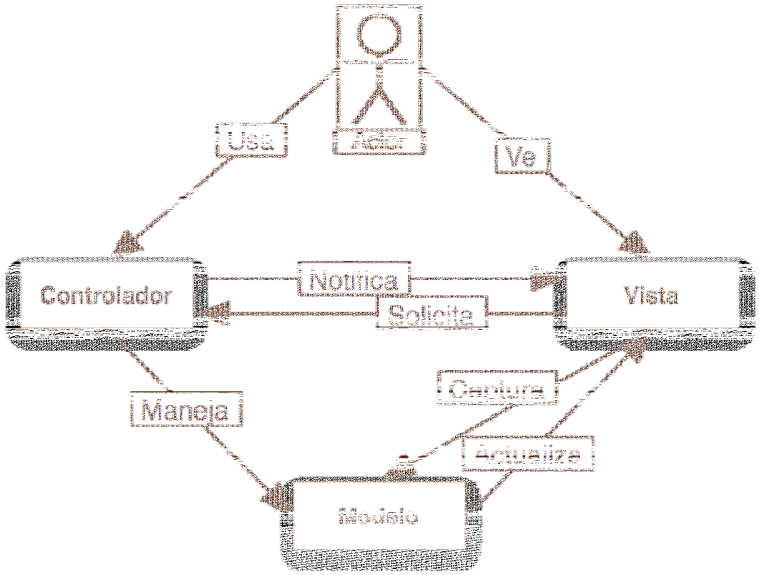
- *Controlador: Maneja los eventos del usuario y actualiza el modelo.*

Es el encargado de percibir cualquier suceso por parte del cliente, cualquier llamada, como por ejemplo un clic en un botón determinado y demás contiene reglas de gestión de eventos.

Para gestionar los datos por lo regular se usa un SGBD (Sistema de Gestión de Base de Datos), por tanto se usaría en el Modelo refiriéndonos a un MVC.

El siguiente diagrama representa la estructura que tiene un MVC:

Figura 2.11: Estructura MVC[15]



Fuente: mvc.mfis.googlepages.com/presentacinmvc.ppt

Explicación:

1. Inicialmente se puede observar que el usuario únicamente interactúa con la interfaz (Vista), por tanto el Usuario/ Actor realiza un evento (pulsar un botón, abrir, actualizar, etc.).
2. La interfaz (Vista) notifica a su respectivo controlador del evento realizado por parte del Usuario. El Controlador entiende y gestiona tal evento.
3. El controlador maneja al Modelo, por tanto puede solicitar cualquier petición a la base de datos como: pedir información, actualizar datos, ingresar datos, eliminar datos, buscar, etc.
4. Luego se despliega nuevamente la Vista, con los cambios visibles para el Usuario obtenidos desde el Modelo.

[15]FUENTE: ***mvc****.mfis.googlepages.com/presentacin****mvc****.ppt*

El modelo no tiene ningún contacto directo con la Vista. Sin embargo, en algunos casos se puede permitir al Modelo notificar cualquier cambio.

5. Vuelve al proceso de espera, donde el Usuario se encuentra habilitado para realizar otro evento

Se entiende como un MVC al modelo activo aquel que SI notifica los cambios en los datos.

Figura 2.12: MVC con modelo Activo

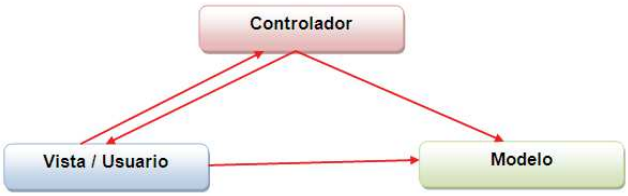
Fuente: www.proactiva-calidad.com/java/patrones/mvc.html

Se puede ver que el Modelo tiene una comunicación unilateral, de tal manera que genera actualizaciones automáticas cada cierto tiempo.

2.3.2.3.1. MVC con un modelo pasivo

Se entiende como un MVC con un modelo pasivo aquel que NO notifica los cambios en los datos.

Figura 2.13: MVC con modelo pasivo[16]



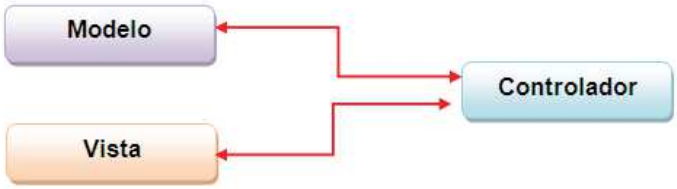
[16] FUENTE: http://www.proactiva-calidad.com/java/patrones/mvc.html

En la ilustración se muestra que el Modelo no genera datos automáticos para la Vista, siempre se conectará mediante el Controlador. Pero sin embargo la Vista podrá pedir datos sin pasar por su controlador.

Desde que se presentó la primera variante de MVC, con el pasar de los años este patrón se ha venido modificando, estancándose en tres variantes fundamentales que se detallan a continuación:

- **Variante 1:**

Figura 2.14: MVC - Pasivo- Variante 1

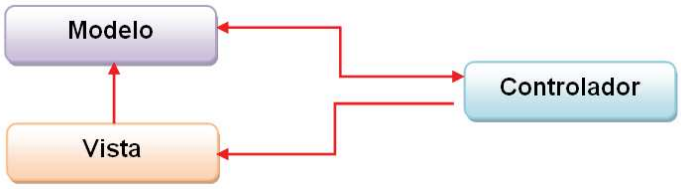


Fuente: www.scribd.com/doc/17677638/mvc

En esta variante la Vista y el Modelo no tienen conexión directa, solamente pueden conectarse a través de su Controlador. Por tanto, la Vista para mostrar los datos del Modelo, deberá primero hacer la petición al Controlador.

- **Variante 2:**

Figura 2.15: MVC – Pasivo – Variante 2

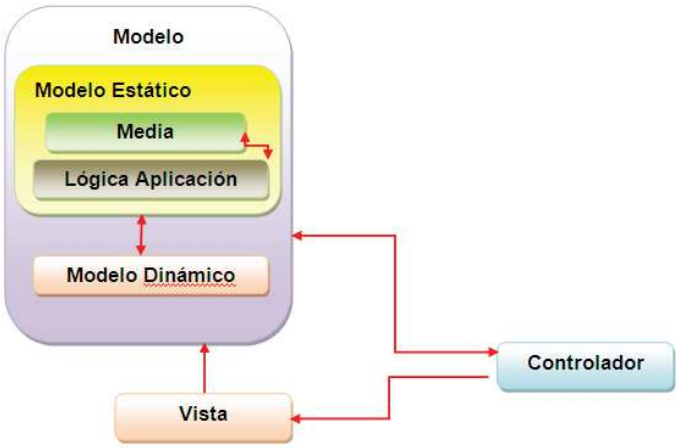


Fuente: www.scribd.com/doc/17677638/mvc

En esta variante ya existe una comunicación entre Vista y Modelo, pero sólo unilateralmente; La Vista pedirá datos directamente en el Modelo dejando así más libertad para el Controlador.

- **Variante 3:**

Figura 2.16: MVC – Pasivo – Variante 3



Fuente: www.scribd.com/doc/17677638/mvc

En esta variante se diversifican las funcionalidades del Modelo, agregando características para nuevas aplicaciones multimedia.

Este patrón de diseño presenta varias ventajas, las más importantes se citan a continuación:[17]

1. Ya que se trabaja en una red, los datos podrían estar cambiando sin que uno se los esté modificando, es posible que otro Usuario los haya modificado. Al usar un modelo activo las Vistas de todos los Usuarios permanecerán siempre actualizadas gracias a las notificaciones que envía el Modelo, por tanto el Programador no debe solucionar el problema de datos inconsistentes.
2. Al estar dividida en varios módulos dentro de su programación, la aplicación se considera modular, brindando código reutilizable.
3. La modificación de cualquier Vista y de cualquier número de Vistas no intervienen, ni afectan a los otros módulos de la aplicación, y de

[17] **Fuente:** http://www.scribd.com/doc/17677638/mvc

igual manera para el Modelo, nada más aumentando o actualizando sus métodos.

4. Actualmente este modelo es muy usado para aplicaciones orientadas a objetos, desarrollados para construir aplicaciones de gran tamaño, puesto que se ha demostrado ser un patrón muy bien elaborado al culminar con aplicaciones que presentan extensibilidad y una mantenibilidad comparadas con otras aplicaciones.

Al igual que cualquier otro patrón de diseño, MVC también presenta algunas desventajas como:

1. Gracias a su diseño muy bien estructurado, el tiempo de desarrollo para cualquier aplicación se extiende, por lo menos en la primera etapa, puesto que MVC necesita que el programador implemente una mayor cantidad de clases, pero sin embargo, también se pudiera considerar una ventaja, ya que al terminar se obtiene una aplicación fácil de mantener, modificar y extender.
2. Es necesario tener las clases Modelo, otra clase Vista y otra Controlador genéricas para poder realizar la comunicación y actualización de manera totalmente transparente para el Usuario.
3. MVC es un modelo que se basa en un lenguaje Orientado a Objetos, por lo que su implementación se tornaría muy difícil en lenguajes que no usen este tipo, por tanto, técnicamente se limitaría para un lenguaje de este tipo.

2.3.2.4. Sistema Gestor de Base de Datos (SGBD).

2.3.2.4.1. Definición SGBD

Una **Base de Datos** es un banco ordenado de datos, el cual permite almacenar información para un uso futuro. Hoy por hoy existe, una gran variedad de Base de Datos de donde escoger, por lo cual se realizará una comparación de las bases de datos más relevantes en el mercado actual.

Un **SGBD** es un sistema para gestionar la información de una base de datos, donde una gestión se describe mediante diferentes acciones como la: inserción, eliminación, actualización, o cualquier otra operación que se pueda realizar a la información.

Los componentes en una base de datos se citan a continuación:

a) **Hardware:** Se refiere a los dispositivos de almacenamiento en donde reside la base de datos, así como a los dispositivos periféricos necesarios para su uso.

b) **Software:** Es el programa que se conoce como SGBD, el cual maneja todas las solicitudes formuladas por los usuarios a la base de datos.

c) **Usuarios:** Existen tres clases de usuarios relacionados con una Base de Datos:

- El Programador de Aplicaciones, quien crea programas de aplicación que utilizan la base de datos.
- El Usuario de la Base de Datos, quien accede a la Base de Datos por medio de un lenguaje de consulta o de programas de aplicación.
- El Administrador de la Base de Datos (DBA Data Base Administrator), quien se encarga del control general del Sistema de Base de Datos.

2.3.2.4.2. Base de Datos Relacionales

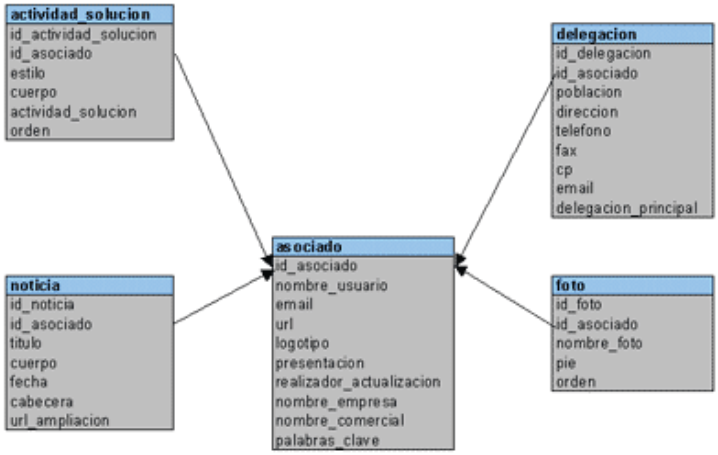
Se construye a través de tablas, registros (líneas), campos (columnas); estás tablas se relacionan por medio de un campo en común en dos o más tablas que tienen las mismas características. A diferencia de una base de datos orientada a objetos, en una relacional existen RDBMS (Sistema de Gestión de Base de Datos Relacional), con el mismo objetivo que la anterior.

Las bases de datos relacionales ya han sido implementadas hace muchos años presentando un buen rendimiento para aplicaciones de negocio o sistemas de

información. Pasan por un proceso al que se le conoce como normalización de una base de datos, el cual es entendido como el proceso necesario para que una base de datos sea utilizada de manera óptima.

Las fortalezas de este tipo de base de datos son: tipos de datos sencillos, lenguajes de consulta potentes, protección elevada; su desventaja sale a la luz cuando sus soluciones no tienen nada que ver con datos de negocio, como: imágenes, multimedia o información geográfica, ya que el modelo relacional no es apropiado por no poseer estructuras de datos para soportar este tipo de datos.

Figura 2.17: Ejemplo de un diseño SQL.[18]



Fuente: www.desarrolloweb.com/articulos/images/img-resupuesto/asociacion/tablas.gif

Existe software exclusivamente dedicado a tratar con bases de datos relacionales. Entre los gestores o manejadores más actuales y populares encontramos: MySQL, PostgreSQL, Oracle, DB2 y Microsoft SQL Server.

Estas descripciones son válidas en general, pero hay que tener en cuenta que algunos sistemas de base de datos no respetan estas fronteras. Por ejemplo, algunos sistemas de base de datos orientados a objetos construidos alrededor

[18]FUENTE:http://www.desarrolloweb.com/articulos/images/img-presupuesto/asociacion/tablas.gif

de lenguajes de programación persistentes se implementan sobre sistemas de base de datos relacionales.

2.3.2.4.3. Base de Datos de Código Cerrado

Por el contrario, el término Código Cerrado, se refiere a que el código fuente es exclusivo de su fabricante, un gran ejemplo es Microsoft®; todos sus códigos fuentes no se encuentran disponibles al público. Por tanto, el software no libre utiliza código cerrado.

La familia de SQL Server, y Oracle son las más importantes SGBD de código cerrado, y serán detalladas a continuación.

2.3.2.4.1. ORACLE

Figura 2.18: Oracle (Logo).

Fuente:www.oracle.com

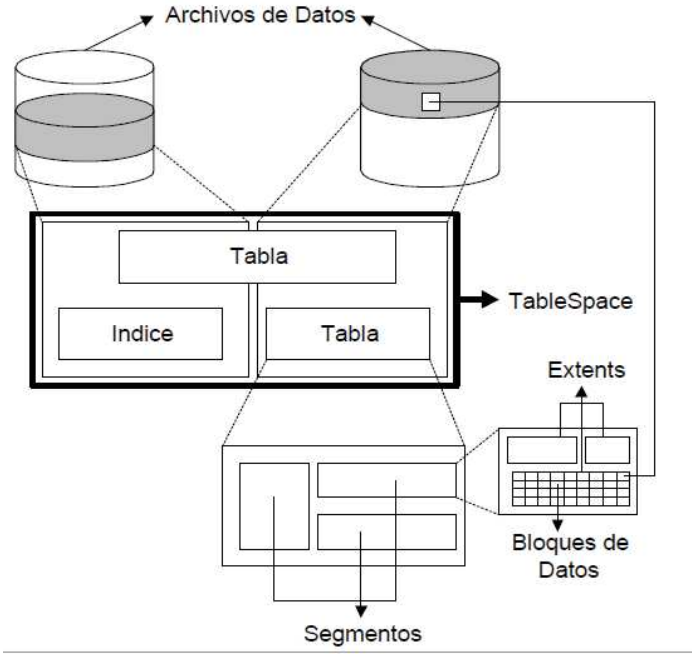
Oracle es un Gestor de Base de Datos muy reconocido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general.

Presenta una estructura relacional o también llamado **RDBMS** por el acrónimo en inglés de Relacional Data Base Management System.

Entre sus principales ventajas se encuentran:

- Soporte de transacciones,
- Estabilidad,
- Escalabilidad y
- Soporte multiplataforma.

Figura 2.19: Estructura RDBMS de Oracle.[19]



Fuente: www.pafumi.net/Arquitectura_Interna_Oracle.pdf.

La Arquitectura general de Oracle consiste de varios procesos corriendo en la máquina donde reside, más la memoria dedicada a ejecutar procesos específicos o al almacenamiento de información de cada proceso y la base de datos física, sus archivos de control, de datos y de transacciones propiamente.

Oracle a partir de la versión 10g Release 2 presenta diferentes versiones que son:

- Oracle Database Enterprise Edition (EE).
- Oracle Database Standard Edition (SE).
- Oracle Database Standard Edition One (SE1).
- Oracle Database Express Edition (XE).
- Oracle Database Personal Edition (PE).
- Oracle Database Lite Edition (LE).

[19]**Fuente:** http://www.pafumi.net/Arquitectura_Interna_Oracle.pdf.

La única edición gratuita es la Express Edition, que es compatible con las demás ediciones de Oracle Database10g R2 y Oracle Database 11g.

Actualmente la última versión en mercado es **Oracle Database 11g,** que trae nuevas ventajas para el usuario.

Es la primera base de datos del mundo en incluir funcionalidades que permiten hacer pruebas de cambios en aplicaciones simulando las cargas reales generadas por los usuarios en los entornos de producción. **Real Application Testing** permite reducir de manera drástica los tiempos, riesgos y costes derivados de la implantación de cambios, asegurando que las aplicaciones se comportarán de manera adecuada y predecible tras las modificaciones. Con esta nueva aplicación los clientes ganan en flexibilidad puesto que pueden responder de manera más efectiva a los requerimientos cambiantes del negocio y hacer una gestión del cambio más efectiva.

2.3.2.4.2. SQL SERVER

Figura 2.20: SQL Server (Logo)

Fuente: www.msqlserver.com

SQL server es un motor de base de datos relacional producido por Microsoft®. El más actual de esta familia es SQL Server 2008 implementa estándares ANSI SQL.

Presenta una gran ventaja al tener una integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business Intelligence Development Studio.

Figura 2.21: Componentes Básicos de SQL Server[20] .

Fuente: www.uaem.mx/posgrado/mcruz/cursos/miic/sqlserver2.ppt

Presenta siete diferentes versiones que son:

- Enterprise Edition (Empresarial).
- Standard Edition (Versión Estándar).
- Workgroup Edition (Para Grupos de Trabajo).
- Express Edition (Edición Personal).
- Compact Edition (Para dispositivos móviles).
- Developer Edition (Para desarrolladores de Software).
- 64-Bit Platform Edition (Para plataformas de 64 Bits).

Tabla 2.6: Características de las diferentes versiones SQL SERVER.

Edición	Beneficio	Tamaño	Características Clave
Express	La forma más rápida de aprender, crear e implementar aplicaciones simples orientadas a	1 CPU 1 gigabyte (GB) RAM 4 GB de tamaño de base de datos	4 GB de tamaño de base de datos Informes simples

[20]FUENTE: *www.uaem.mx/posgrado/mcruz/cursos/miic/sqlserver2.ppt*

Edición	Beneficio	Tamaño	Características Clave
	datos.		
Workgroup	Solución de base de datos sumamente accesible y muy fácil de usar para pequeños departamentos y empresas en crecimiento.	1 o 2 CPU 3 GB RAM	Management Studio Importación/Exportación Organización en clústeres Transmisión de registros de seguridad
Standard	Plataforma completa de administración y análisis de datos para empresas medianas y grandes departamentos.	1 a 4 CPU RAM ilimitada	Reflejo de bases de datos Estándar con Análisis Servicios Informes estándar con Reporting Servicies Réplica completa y Disponible en ediciones nativas de 32 y 64 bits Admite Itanium 2 y x64
Empresarial	Plataforma totalmente integrada de administración y análisis de datos para aplicaciones empresariales esenciales.	Escala y particionamiento ilimitados	Reflejo de bases de datos avanzados, operaciones completas en línea y paralelas Creación avanzada de informes con informes especiales, de muy alto nivel y personalizados Enrutamiento de datos y

Edición	Beneficio	Tamaño	Características Clave
			capacidades de transformación Disponible en ediciones de 32 y 64 bits Admite Itanium 2 y x64

Fuente: *www.uaem.mx/posgrado/mcruz/cursos/miic/sqlserver2.ppt*

2.3.2.4.3. Comparación entre los principales SGBD

Las comparaciones se realizan con las versiones más actuales de cada SGBD, de la siguiente manera:

- **MySQL 5.1**
- **PostgreSQL/8.1.3**
- **Microsoft SQL Server 2008**
- **Oracle 11g** (Sin embargo, la compañía dice que es probable que tarde al menos dos o tres años en comenzar a migrar a la versión 11g desde el programa anterior, 10g. Por tanto, las comparaciones se realizarán en la versión 10g).

Tabla 2.7: Comparación de los principales SGBD.

Parámetro	Mysql	Postgresql	SQL Server	Oracle
Velocidad y Robustez	X	X	X	X
GNU (Sistema Libre)	X	X		
Versión Express			X	X
Tablas B-Tree	X			
Multiproceso	X	X	X	X
Multi-Hilo	X	X	X	X
Varios Sistemas Operativos	X	X		
Sistema de Contraseñas y	X	X	X	X

Parámetro	Mysql	Postgresql	SQL Server	Oracle
Privilegios				
Conexión TCP/ IP	X	X	X	X
Trabajo con Concurrencia	X	X	X	X
Generación de Nuevos Tipos de Datos		X		
Usa Estándar SQL	X	X	X	X
Sencillez de Trabajo (Según la comunidad)	X	X	X	
Compatibilidad con Active Directory			X	
Compatibilidad con la autenticación de Windows.			X	
Conexión Directa con Visual Studio			X	
Contabilidad con XML		X	X	X
Acogida en el Mercado			X	X
Apoyo por la Comunidad	X	X		
Integridad Referencial	X	X	X	X
Cifrado de Información	X	X	X	X

Fuente: Autores

2.3.2.4.4. SQL SERVER 2008 en Covial

En el siguiente cuadro consta que el gestor de bases escogido cumple a cabalidad con las necesidades de la empresa.

Tabla 2.8: SQL SERVER 2008 EN COVIAL

Concurrencia	SQL SERVER presenta buena respuesta a concurrencia puesto que mientras está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla.
Respaldos	Actualmente existen una serie de comandos que permite hacer una copia de seguridad de una o múltiples bases de datos.

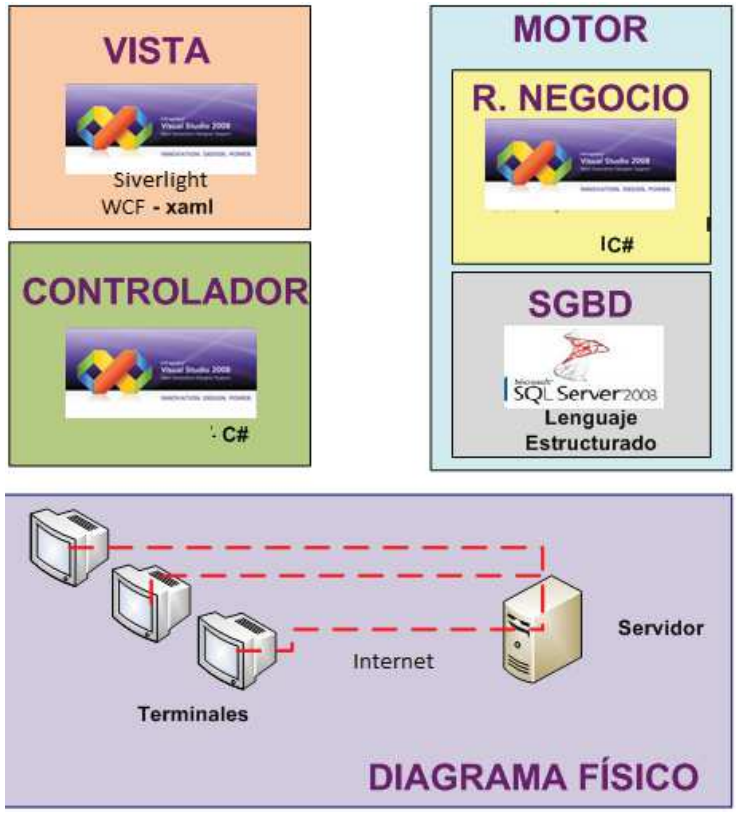
	Hay varias formas de realizar un backup en SQL, desde su gestor o en algunos casos con la instalación previa de un programa auxiliar. De esta manera se tendrá siempre un respaldo de todos los datos que se haya ingresado en la base de datos, brindando seguridad e integridad.
Procedimientos almacenados , triggers y funciones	Se empleará procedimientos almacenados y funciones en COVIAL para posibles acciones de: ingreso, búsqueda, eliminación o actualización. Los procedimientos almacenados mejoran el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente.
Compatibilidad con .NET	Es compatible con el API Visual Studio 2008.
Soporte y Documentación	Gracias a que SQL es un lenguaje muy maduro, tanto en libros y en la Internet se puede encontrar información abundante sobre este gestor. Si se llegará a tener alguna dificultad se podrá recurrir a cualquiera de estos para solucionarlos.
Cuentas de Usuario	Es importante tener una buena administración de la base de datos brindando una mayor seguridad de la información. Las cuentas de usuario están ligadas a los diferentes niveles de permisos.
Seguridad	Almacenar contraseñas en SQL como texto plano (sin encriptar) nunca será una buena idea. Como buenos administradores de bases de datos se debe tener cuidado de proteger la información de nuestros usuarios. Afortunadamente SQL provee diversas opciones para proteger contraseñas. Es posible usar SHA1 y MD5. Y aunque se han encontrado formas de romper contraseñas que utilizan estos métodos de encriptación, siempre será mejor tener esta opción a la

	de contraseñas en texto plano.
Integridad de Datos	COVIAL requiere que los reportes generados muestren datos reales y actuales, por tanto, la integridad pasa a ser un requisito para la empresa. Se puede decir de manera simple que integridad referencial significa que cuando un registro en una tabla haga referencia a un registro en otra tabla, el registro correspondiente debe existir SQL presenta un gran número de herramientas para garantizar la integridad de los datos como: claves primarias y foráneas, claves únicas, etc.

Fuente: ***Autores***

2.3.3. DEFINICIÓN DE HERRAMIENTAS EN WEB COVIAL 1.0

Figura 2.22 Diagrama WEB COVIAL 1.0



Fuente: Autores

WEB COVIAL 1.0 seguirá las siguientes características:

- WEB COVIAL 1.0 se aplicará dentro de una arquitectura Cliente-Servidor, donde cada uno de los Clientes se encuentran accediendo al sistema a través de la inmensa Internet; mientras que el Servidor, el cual se conecta el cliente, responde todas sus peticiones tanto como su carga sea posible.
- El nuevo Sistema se encuentra realizado bajo un modelo MVC (MODELO – VISTA-CONTROLADOR), entendiendo que este modelo es también conocido como una variación a la arquitectura en capas.

  MODELO: Se encuentra el repositorio de Datos, WEB COVIAL 1.0 realizado con SQL SERVER 2008, utilizando sus grandes ventajas.

  Integra además un conjunto de reglas definiendo la lógica del negocio mediante el lenguaje C#.

  VISTA: Programado para una interfaz genuina e intuitiva para el usuario en un ambiente WEB. Utilizando herramientas Microsoft tal como SilverLight para GUI y Windows Comunication Foundation (WCF) para una conexión mediante servicios.

  Además de utilizar C# para su programación de Objetos.

  Es importante recordar que es necesaria la previa instalación del Framework 3.5 o superior.

  CONTROLADOR: Es también llamado un controlador de Eventos. En palabras sencillas son los eventos de la interfaz que usa el usuario, tal como, CLICK, DOUBLE_CLICK, CHANGED, etc.

  Además de utilizar C# para su programación de Objetos.

  NOTA: WEB COVIAL 1.0 divide en dos archivos la Vista y sus Eventos, añadiendo más seguridad al sistema.

3. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

3.1. ANÁLISIS DE REQUERIMIENTOS.

3.1.1. ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE.

Para la determinación de los requerimientos de software, SRS por sus siglas en inglés, se seguirá el estándar *IEEE Std 830-1998: "IEEE Recommended Practice for Software Requirements Specifications"* (Prácticas Recomendadas para las Especificaciones de Requerimientos de Software), publicado el 25 de junio de 1998, que es una revisión del *IEEE Std 830-1993,* actualmente en estado archivado y reemplazado por el más reciente.

Las definiciones que no se establezcan aquí, serán tomadas de las que proporciona el estándar *IEEE Sed 610.12-1990: "IEEE estándar glosar of software engineering terminology"* (Glosario estándar de terminología de ingeniería de software), publicado el 10 de diciembre de 1990, que se encuentra en estado activo.

3.1.1.1. Introducción (Sección 1 de la SRS).

3.1.1.1.1. Justificación:

En la actualidad la fundación COVIAL no tiene un sistema que maneje la información o interactúe con sus usuarios de forma remota mediante el Internet. Los usuarios de la fundación tienen que acceder a los servicios brindados por COVIAL de forma presencial, lo cual causa poca concurrencia de parte de los afectados. La fundación COVIAL en la actualidad maneja la información de manera escrita, razón por la que su recopilación es lenta y sus reportes son inexactos y demorados.

Este sistema de control de información Web brindará una mejor organización dentro de la fundación, mejor análisis de todos los incidentes de tránsito y un mejor manejo de noticias en el cual podrán publicar sus nuevos eventos. El portal también permitirá charlas o acontecimientos que ocurren durante el mes.

Los usuarios podrán aclarar sus dudas sobre temas de tránsito, y se realizará servicios de comunicación directa entre afectados y los expertos de COVIAL.

3.1.1.1.2. Alcance

El sistema contemplará para su desarrollo, principalmente, los siguientes Objetivos del Negocio (ON), Criterios de Éxito (CE) y Riesgos del Negocio (RN):

ON-1: El sistema de manejo de información Web constará de páginas interactivas e informativas con sus respectivas animaciones para brindar una experiencia multimedia al usuario.

ON-2: Incluirá módulos de capacitación remota en base a catálogos virtuales en 3-D, y videos informativos.

ON-3: Todas estas páginas y controles serán desarrolladas en aspx utilizando tecnología .Net, Microsoft Silverlight y xaml manipuladas con WFC o Windows Comunication Foundation y Expresiones Studio.

ON-4: El portal se conectará a una base de datos creada en SQL Server 2008 que manejará la información de la fundación y se conectará mediante Microsoft Enterprise Library 4.1.

ON-5: Incluirá un foro en donde los afectados podrán ingresar sus quejas, dudas, reclamos o acontecimientos, el cual constará con un módulo de filtrado en el que los moderadores podrán responder y administrar todos los mensajes recibidos.

ON-6: En el portal figurará un chat en línea desarrollado con la unión de las tecnologías de Silverlight y Ajax, con esto los usuarios podrán comunicarse en tiempo real con los expertos.

ON-7: El portal ofrece diversos juegos que serán accesibles siempre y cuando se haya aprobado una pequeña prueba en base a Educación Vial, y así concientizar a las personas desde corta edad.
También se podrá publicar folletos, logrando así que las personas se interesen en leer este tipo de información, y tener módulos de educación virtual.

ON-8: El sistema mostrará estadísticas y reportes desarrollados en .Net. Este módulo también desplegará noticias y eventos relacionados a la fundación, estos podrán ser administrados por lo cual se podrán añadir, editar y eliminar noticias.

CE-1: Tener por lo menos un 75% de los clientes, que acceden actualmente en COVIAL de manera presencial, utilizando el sistema, en los próximos 3 meses desde la implementación en COVIAL de la primera versión del software.

Métrica: Número de usuarios y accesos al sistema.

RN-1: La implementación del software puede generar nuevos roles que tienen que ser asumidos por el usuario.

Probabilidad = 0.25; *Impacto* = 2/10.

RN-2: Algunos usuarios pueden no adaptarse al nuevo sistema dentro del tiempo especificado en los Objetivos del Negocio.

Probabilidad =0.25; *Impacto* = 8/10.

NOTA: Los datos de “probabilidad” e “impacto” se obtuvieron directamente de COVIAL en entrevista directa a la Gerencia.

WEB COVIAL 1.0 incluirá las siguientes Características Principales (CP):

CP-1: Realizado en Visual Studio .NET 2008, que incorporará en un mismo entorno todos los componentes necesarios para el desarrollo de aplicaciones de negocios de máxima calidad.

CP-2: Sistema Multi-capas, **WEB COVIAL 1.0** estará diseñado en una arquitectura con tecnología abierta, formada por múltiples capas lógicas e independientes entre sí, esto permitirá tener una arquitectura completamente versátil sobre la cual se puede incluir en cualquier momento sistemas avanzados de ejecución entre capas, por ejemplo: sistemas de lectura, sistemas de seguridad, capas de filtrado de información, etc.

CP-3: Código Fuente C# 2.0, lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Que ha recogido las mejores características de sus antecedentes (C, C++, Java, Delphi, Smalltalk, Pascal) para obtener una sintaxis versátil y sencilla que permite optimizar el tiempo de desarrollo al máximo.

CP-4: WEB COVIAL 1.0 hace uso de un eficaz monitor transaccional que permite entre otras cosas un aprovechamiento óptimo de la capacidad de procesamiento y almacenamiento de información de él o los Servidores; almacenamiento y reverso efectivo de operaciones encerradas en un mismo proceso operativo de manera de que no se almacenen transacciones incompletas o inconsistentes.

CP-5: Base de Datos Relacional SQL Server Express Edición 2008 o SQL Server 2008, que ofrece mayor escalabilidad, disponibilidad y seguridad a la información empresarial y las aplicaciones de análisis al tiempo que simplifica su creación, implantación y gestión.

CP-6: En la Web, las aplicaciones tienen más representatividad y, por ende, su atractivo y su facilidad de uso necesitan potenciarse. Silverlight 2.0 con herramientas muy poderosas para brindar al usuario una interfaz agradable y amigable, ayudándose de un lenguaje XAML.

CP-7: La Interfaz Servidor-Cliente mediante WINDOWS COMUNICATION FOUNDATION comprime al máximo la información que viaja desde y hacia el servidor, ya que se trata únicamente de información útil, a diferencia de los sistemas Web Enabled en donde además de los datos propios del sistema viajan páginas Web, complementos adicionales (Active X, Animaciones, Imágenes, etc.) y otros, que requieren de un mayor ancho de banda, por lo que sin duda la arquitectura de **WEB COVIAL 1.0** optimiza al máximo las líneas de comunicaciones existentes, lo que le permite a la Institución el uso de cualquier tipo de red disponible en el mercado que se ajuste con presupuestos y capacidad.

CP-8: Sistemas de seguridad de última tecnología; WCF permite crear servicios Web seguros; Encriptamiento de claves, cadenas de conexión y cadenas de control en los registros de la base de datos para evitar que los datos sean cambiados por terceros, Seguridad en Niveles Integrada en SQL Server 2008 permitiendo privacidad de los datos a alto nivel.

CP-9: Por sus características técnicas **WEB COVIAL 1.0** no requiere de Servidor(es) de gran tamaño, permitiendo que la plataforma tecnológica (Hardware) crezca a la par de las condiciones de mercado.

- **CP-10: WEB COVIAL 1.0** corre con total fiabilidad en equipos con sistemas operativos: Windows 2000, Windows XP, Windows 2003 y 2008 server, Windows 7.

3.1.1.1.3. Definiciones, Siglas y Abreviaciones:

a) **DALC:** Data Access Layer Component (Capa de Acceso a Datos)
b) **MDI:** Multiple Document Interface (Interfaz de Múltiples Documentos).
c) **OMG:** Object Management Group.
d) **SDI:** Single Document Interface (Interfaz de Documento Único).
e) **SO:** Sistema Operativo.
f) **SRS:** Software Requirements Specifications (Especificaciones de Requerimientos de Software).
g) **TBD:** To Be Defined (Ha ser definido).
h) **UML:** Unified Modeling Language (Lenguaje Unificado de Modelado).
i) **WAN:** Wide Area Network (Red de área extendida).
j) **Windows:** Sistema Operativo Microsoft® Windows.
k) **WCF:** Windows Comunication Foundation.
l) **XAML:** Lenguaje funcional de Silverlight y WPF

3.1.1.1.4. Descripción Global (Sección 2 de la SRS):

3.1.1.1.4.1. Perspectiva del Producto:

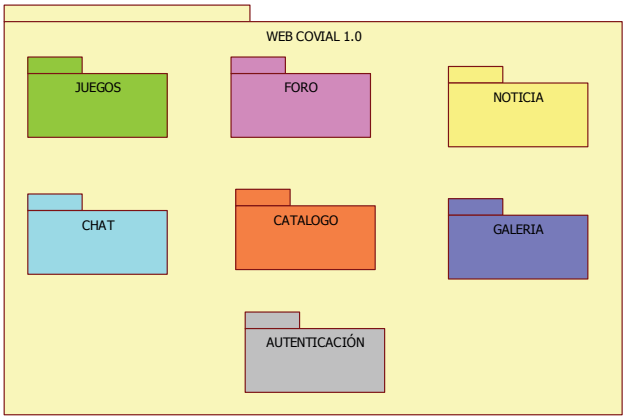
WEB COVIAL 1.0 es un sistema orientado a satisfacer a los clientes de COVIAL. Presentará una interfaz amigable al usuario, de fácil aprendizaje, y que proporcionará mecanismos para la culminación rápida y correcta de los procesos.

WEB COVIAL 1.0 será un producto independiente y autónomo que trabajará en plataformas basadas en Windows. No tiene interrelación con otras soluciones de software existentes en COVIAL**.**

Para definir los componentes del sistema, se seguirá UML 2.2 sobre los Casos de Uso, como está definida en el Documento OMG Número: **formal/2009-02-02**[21].

El diagrama está diseñado con paquetes representados por rectángulos en forma de carpeta. Dentro de cada paquete se encuentran los casos de uso agrupados según su similitud y funcionalidad. Un paquete representa un módulo a implementar, y se separan por colores según los perfiles de usuario o los roles. Existen paquetes que pueden ser accedidos por varios actores.

Figura 3.1: Diagrama de Funcionalidades WEB COVIAL 1.0



Fuentes: Autores

3.1.1.1.4.2. Interfaces del Sistema.

WEB COVIAL 1.0 contendrá tres interfaces para la interacción con los diferentes módulos, correspondientes a cada una de las tres capas en las que se programará:

a) **Capa de Presentación:** Una GUI para la gestión de la información, basada en pestañas. (Programación Cliente)

[21]"OMG Unified Modeling Language™ (OMGUML), Superstructure"*Version 2.2.*

b) **Capa de Aplicación:** Una librería que contenga las clases e implemente reglas del negocio. (Programación librerías y Proxies)

c) **Capa de Datos:** Una librería que contenga la DAL (Librería de Acceso de Datos) para la comunicación entre la capa de aplicación y la base de datos. (Base de datos)

WEB COVIAL 1.0 no contempla la unificación con otros sistemas y, por lo tanto, no provee interfaces para que sistemas de terceros se acoplen de manera alguna.

3.1.1.1.4.3. Interfaz con el Usuario.

El usuario accederá a la aplicación por medio de una GUI, que contendrá una ventana principal y varias pestañas secundarias. Se desarrollará un GUI WEB que sea intuitiva, amigable, y con acceso rápido a las funciones más comunes; para obtener el máximo rendimiento y reduciendo el tiempo necesario para la realización de las tareas.

3.1.1.1.4.4. Interfaz con el Hardware.

WEB COVIAL 1.0 tendrá las siguientes interfaces con el hardware:

- **De entrada:** Ratón y Teclado.
- **De salida:** Monitor.

Todas las interfaces estarán siempre activas para el funcionamiento del software, y requerirá de todas para su funcionamiento.

3.1.1.1.4.5. Interfaz con el Software.

Para realizar la interconexión con el Sistema Operativo y la base de datos se emplearán bibliotecas de terceros. Se determinará el software más adecuado y las librerías necesarias dependiendo del software seleccionado, según los siguientes criterios:

TBD-1: Para el Sistema Operativo.- Un Framework que sirva de soporte a la aplicación a desarrollarse, procurando hacer el software independiente de la versión del SO.

TBD-2: Para la Base de Datos.- Un Conector (de ser necesario), para la comunicación entre la aplicación y la base de datos.

3.6. USO DE METODOLOGÍA.

3.6.1. DIAGRAMAS DE CASOS DE USO

Es una técnica para capturar información sobre los servicios que un sistema proporcional a su entorno, desde el punto de vista del usuario. Especifica los requisitos independientemente de la implementación, definiendo los límites del sistema y las relaciones dentro el sistema y el entorno.

Es importante recordar que los Casos de Uso se realizan desde la visión de la Clase de Usuario, y no de los programadores.

Al modelar el Diagrama de Casos de Uso no se busca analizar el detalle, y mucho menos los flujos. Todo ese pormenor se lo podrá plasmar en otro tipo de diagramas, como los diagramas de secuencia, de estados, o simplemente un texto en una especificación.

Hay que tomar en cuenta que entre estos diagramas de uso se encuentran dos tipos de relaciones:

- **<<include>>**En términos muy simples, cuando se relacionan dos casos de uso con un “include”, se dice que el primero (el caso de uso base) incluye al segundo (el caso de uso incluido). Es decir, el segundo es parte esencial del primero. Sin el segundo, el primero no podría funcionar bien pues no podría cumplir su objetivo.
- **<<extend>>**Una de las diferencias básicas es que en el caso del “extend” hay situaciones en que el caso de uso de extensión **no es indispensable que ocurra**, y cuando lo hace ofrece un valor extra (extiende) al objetivo original del caso de uso base. En cambio en el

"include" es necesario que ocurra el caso incluido, tan sólo para satisfacer el objetivo del caso de uso base.

El objetivo de estos tipos de relaciones **NO** consiste en motivar la división de los casos de uso en la mayor cantidad de pedazos, pues esto enreda el diagrama. Entendiendo esto y siendo congruentes, se obtendrá un beneficio real para el proyecto.

3.6.1.1. Requerimientos e Identificación de Actores

La primera etapa de RUP es la obtención de Requerimientos, sin embargo no se presentan ya que el **Capítulo 1** se centra en este tema.

Tipos de Actores:

- PRINCIPALES: Personas que usan el sistema.
- SECUNDARIOS: Personas que mantienen o administran el sistema.
- MATERIAL EXTERNO: Dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- OTROS SISTEMAS: Sistemas con los que el sistema interactúa.

Tabla 3.1: Actores UML en WEBCOVIAL 1.0

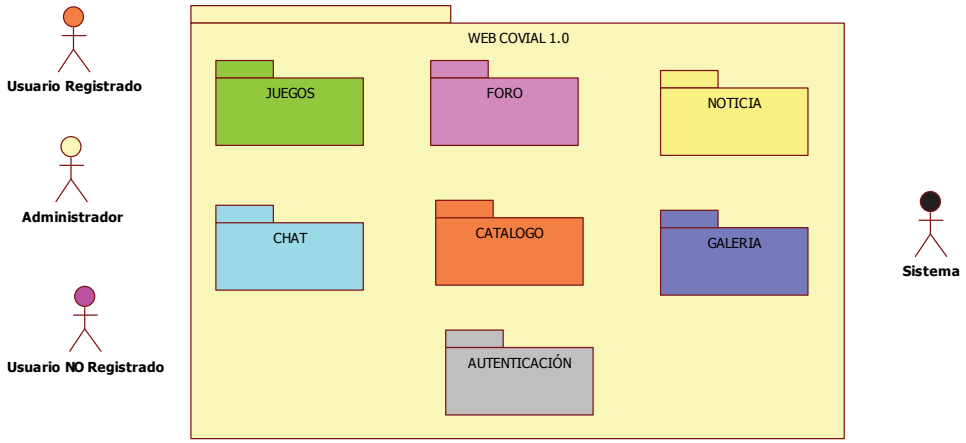
ACTORES EN WEB COVIAL 1.0	
Usuario COVIAL (Visitante de la página WEB)	Actor principal (Usuario Final).
Usuario COVIAL (Visitante de la página WEB) REGISTRADO	Actor principal (Usuario Final).
Administrador	Actor Principal.(Programadores)
Base de Datos	Sistema. SGBD que forman parte del ámbito de la aplicación y debe ser utilizado.

Fuente: Autores

3.6.1.2. Identificación de Casos de Uso.

3.6.1.2.1. Caso de Uso Global

Figura 3.2: Caso de uso Global COVIAL.

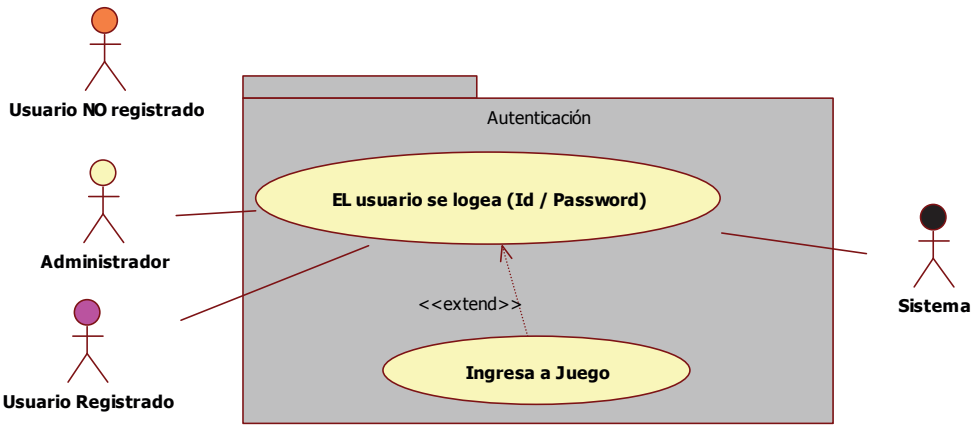


Fuente: Autores

La Figura representa el Caso de Uso de COVIAL de la manera global (paquete) con su comportamiento.

3.6.1.2.1. Sub Caso de Uso Autenticación

Figura 3.3: Sub-Caso de Uso Autenticación.



Fuente: Autores

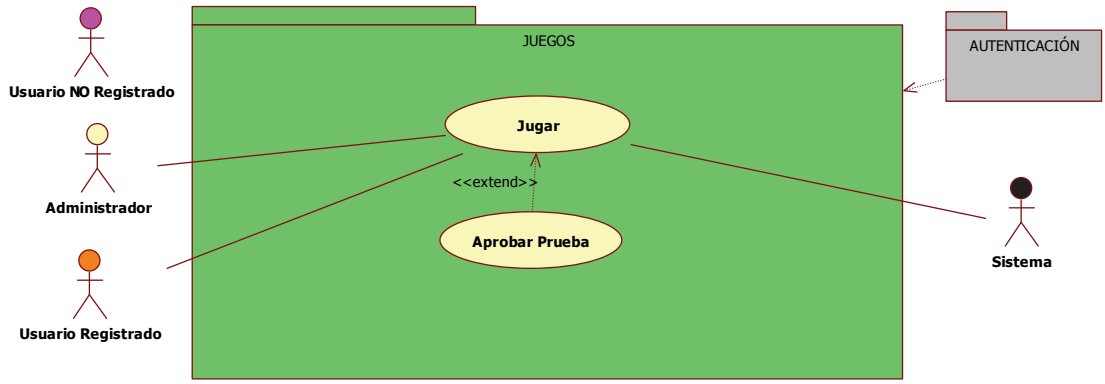
Tabla 3.2: Descripción Sub-Caso de Uso Autenticación

DESCRIPCIÓN SUB-CASO AUTENTICACIÓN	
ACTORES	Usuario Registrado, Administrador.
OBJETIVO	Ingresar al Sistema WEB COVIAL 1.0.
DESCRIPCIÓN	El usuario digita tanto su clave como su contraseña. El sistema valido la información.
PRECONDICIÓN	Ninguna.
ALTERNATIVA	Ninguna.
EXTENSIONES	Si no han ocurrido errores, la opción se genera correctamente. Caso contrario, el sistema pedirá verificar los datos ingresados (Posibles errores humanos), enviando el mensaje correspondiente.

Fuente: Autores

3.6.1.2.2. Sub Caso de Uso Juegos

Figura 3.4: Sub-Caso de Uso Juegos.



Fuente: Autores

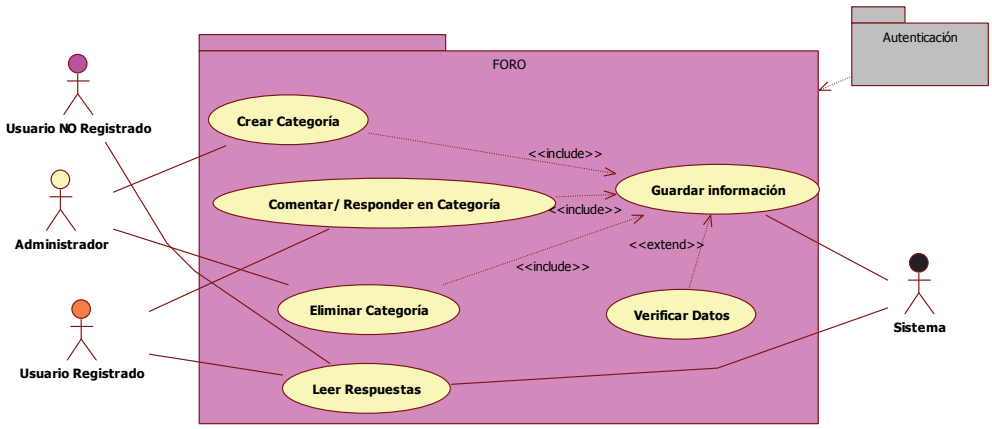
Tabla 3.3: Descripción Sub-Caso de Uso Juegos

DESCRIPCIÓN SUB-CASO JUEGOS	
ACTORES	Usuario Registrado, Administrador.
OBJETIVO	Ingresar a Juegos Definidos en el Sistema.
DESCRIPCIÓN	El usuario ingresa al Módulo de juegos. El usuario NO REGISTRADO no tiene acceso al módulo
PRECONDICIÓN	Previamente Autenticado siendo: ADMINISTRADOR o USUARIO REGISTRADO.
ALTERNATIVA	Ninguna.
EXTENSIONES	Si el usuario pasa la prueba COVIAL-CONOCIMIENTOS puede ingresar a juegos Caso contrario, el sistema pedirá que rinda la prueba en otro momento.

Fuente: Autores

3.6.1.2.3. Sub Caso de Uso Foro

Figura 3.5: Sub-Caso de Uso FORO.



Fuente: Autores

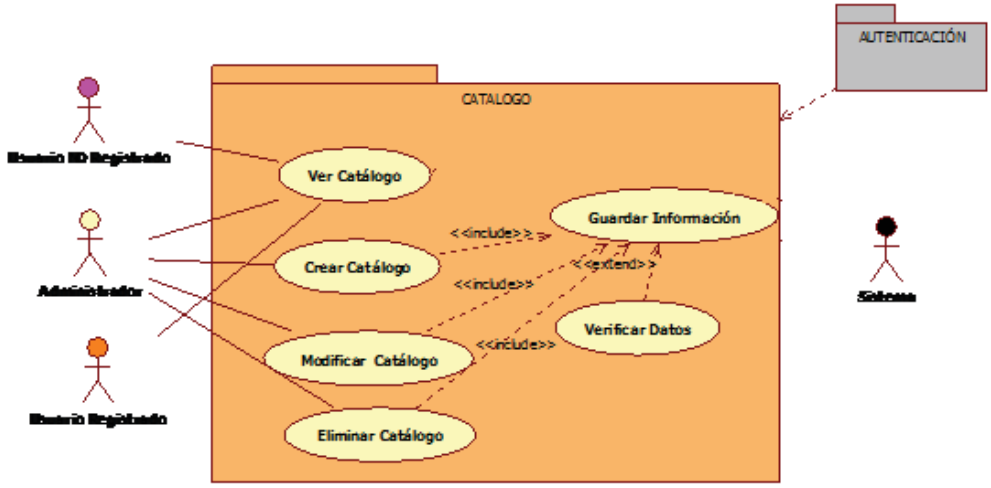
Tabla 3.4: Descripción Sub-Caso de Uso Foro.

DESCRIPCIÓN SUB-CASO FOROS	
ACTORES	Usuario Registrado, Administrador, Usuario NO Registrado.
OBJETIVO	Ingresar e interactuar con el Foro del Sistema.
DESCRIPCIÓN	EL usuario interactúa con el FORO
PRECONDICIÓN	Previamente Autenticado siendo: ADMINISTRADOR o USUARIO REGISTRADO.
ALTERNATIVA	Ninguna.
EXTENSIONES	Antes de que el Administrador pueda crear o modificar categorías el Sistema Validará la información.

Fuente: Autores

3.6.1.2.4. Sub Caso de Uso Catálogo

Figura 3.6: Sub-Caso de Uso Catálogo.



Fuente: Autores

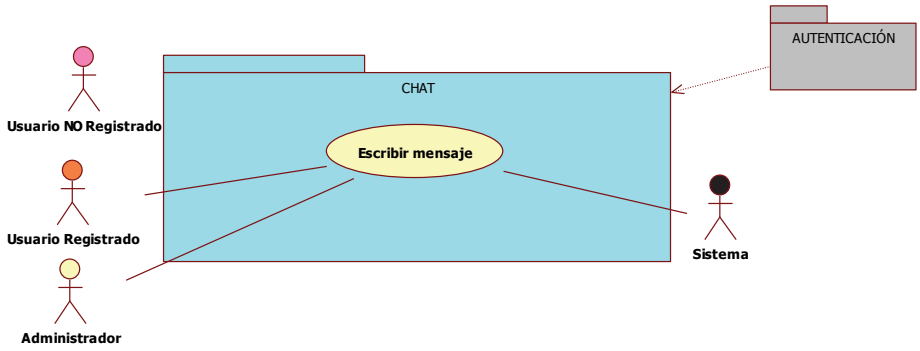
Tabla 3.5: Descripción Sub-Caso de Uso Catálogo.

DESCRIPCIÓN SUB-CASO CATÁLOGO	
ACTORES	Usuario Registrado, Administrador, Usuario NO Registrado.
OBJETIVO	Ingresar e interactuar con el Foro del Sistema.
DESCRIPCIÓN	EL usuario interactúa con el Catálogo.
PRECONDICIÓN	Previamente Autenticado siendo: ADMINISTRADOR o USUARIO REGISTRADO.
ALTERNATIVA	Ninguna.
EXTENSIONES	Antes de que el Administrador pueda crear o modificar catálogos el Sistema Validará la información.

Fuente: Autores

3.6.1.2.5. Sub Caso de Uso Chat

Figura 3.7: Sub-Caso de Uso Chat.



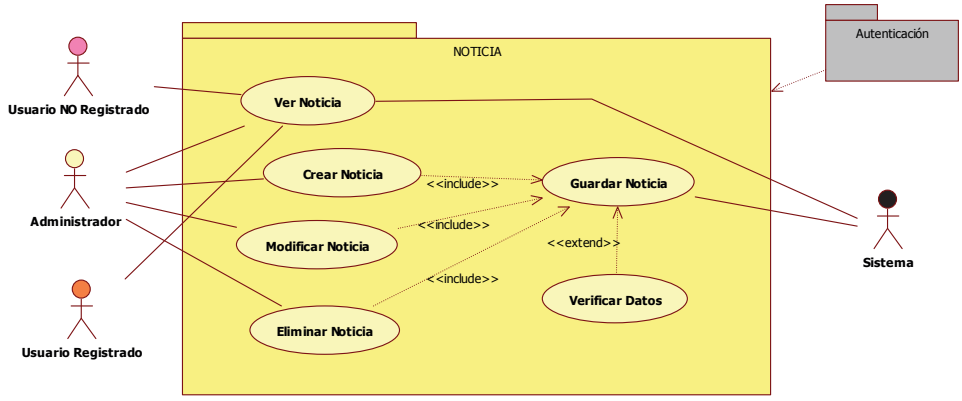
Fuente: Autores

Tabla 3.6: Descripción Sub-Caso de Uso Catálogo

DESCRIPCIÓN SUB-CASO CATÁLOGO	
ACTORES	Usuario Registrado, Administrador.
OBJETIVO	Interactuar entre usuarios registrados en un Chat.
DESCRIPCIÓN	EL usuario interactúa con las personas en un Chat.
PRECONDICIÓN	Previamente Autenticado siendo: ADMINISTRADOR o USUARIO REGISTRADO.
ALTERNATIVA	Ninguna.
EXTENSIONES	Ninguna.

3.6.1.2.6. Sub Caso de Uso Noticia

Figura 3.8: Sub-Caso de Uso Noticia.



Fuente: Autores

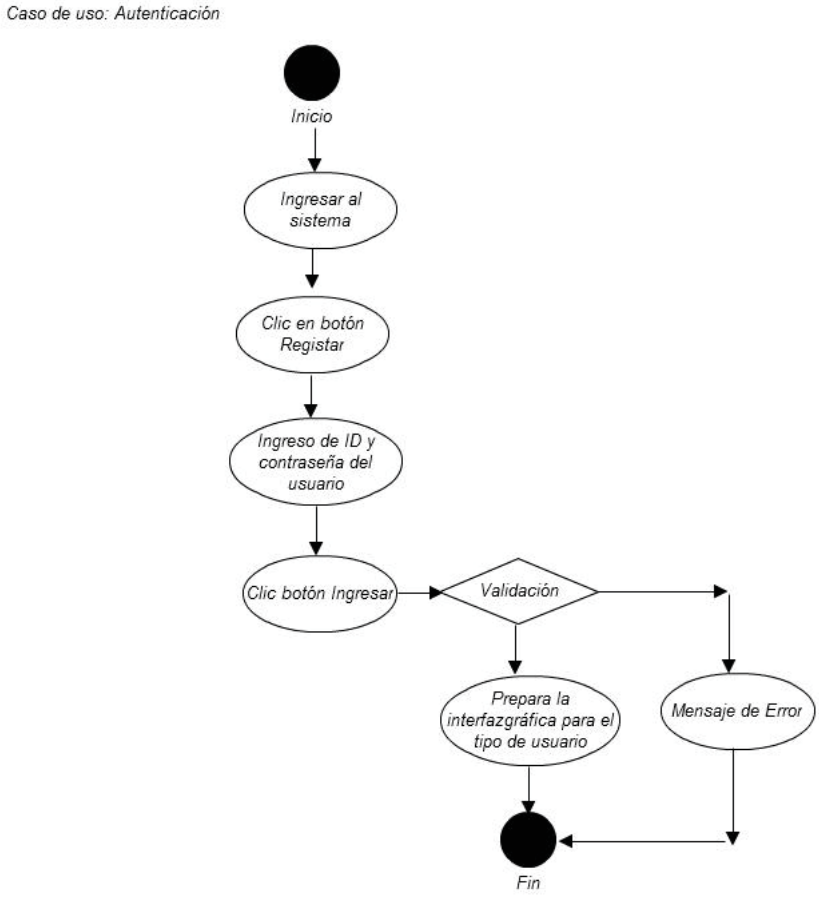
Tabla 3.7: Descripción Sub-Caso de Uso Noticia

DESCRIPCIÓN SUB-CASO NOTICIA	
ACTORES	Usuario Registrado, Administrador, Usuario NO Registrado.
OBJETIVO	Dar a conocer las noticias importantes a los diferentes Usuarios.
DESCRIPCIÓN	EL usuario interactúa con las personas en un Chat.
PRECONDICIÓN	Previamente Autenticado siendo: ADMINISTRADOR o USUARIO REGISTRADO.
ALTERNATIVA	Ninguna.
EXTENSIONES	Antes de que el Administrador pueda crear, modificar o eliminar Noticias el Sistema Validará la información.

Fuente: Autores

3.6.2. DIAGRAMAS DE ACTIVIDADES

Figura 3.9: Diagrama de actividad Caso de Uso Autenticación

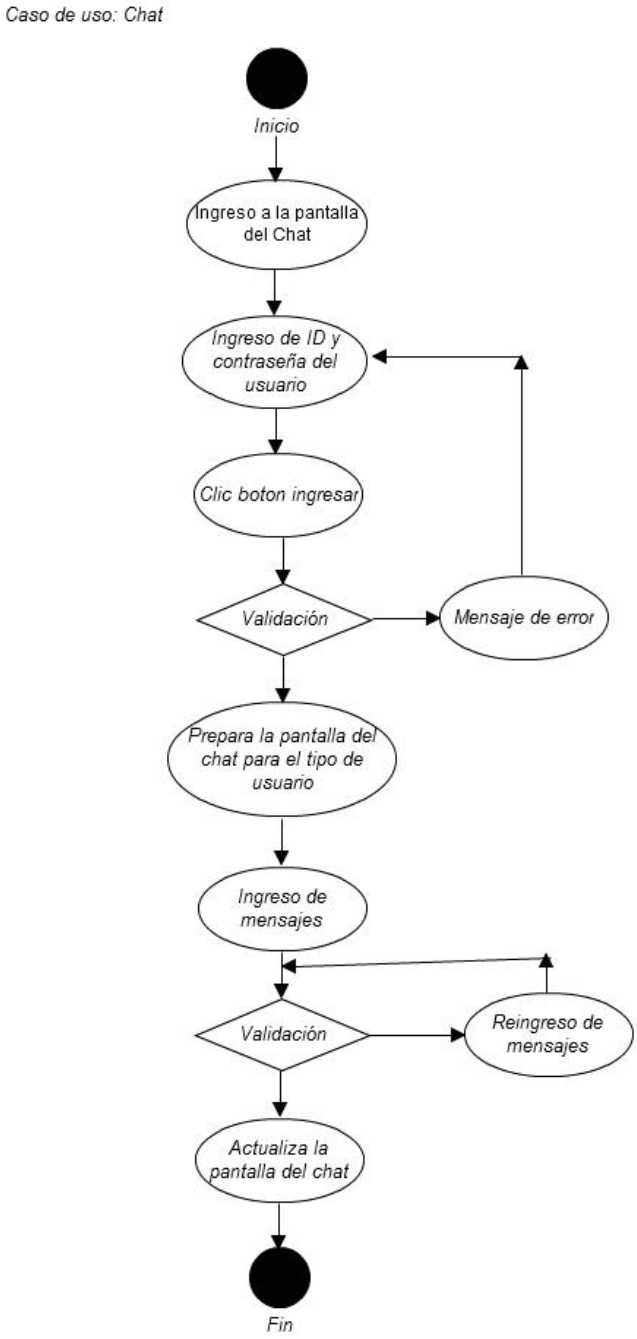


Fuente: Autores

Figura 3.10: Diagrama de actividad Caso de Uso Catalogo

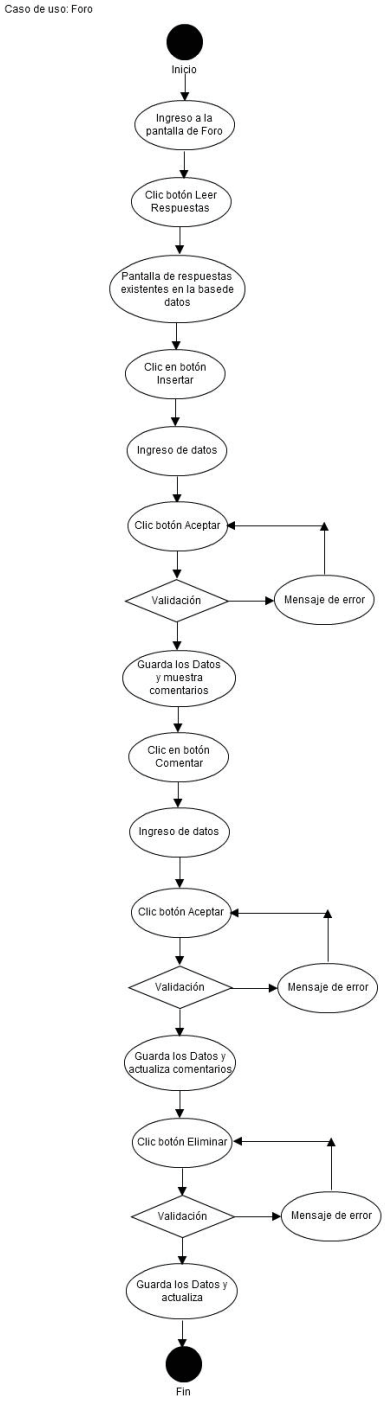
Fuente: Autores

Figura 3.11: Diagrama de actividad Caso de Uso Chat



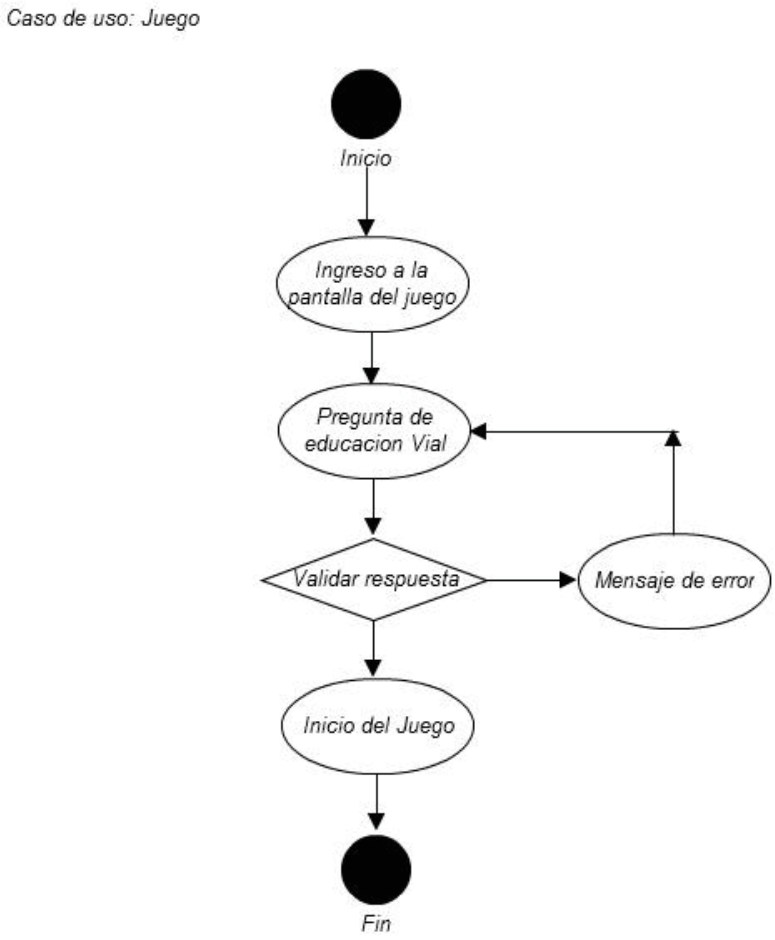
Fuente: Autores

Figura 3.12: Diagrama de actividad Caso de Uso Foro



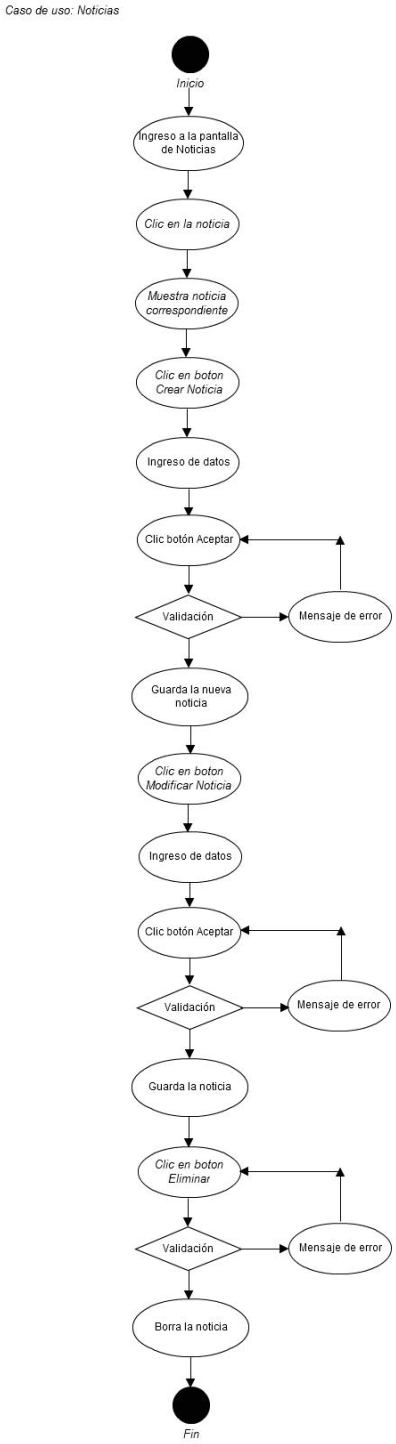
Fuente: Autores

Figura 3.13: Diagrama de actividad Caso de Uso Juego



Fuente: Autores

Figura 3.14: Diagrama de actividad Caso de Uso Noticias



Fuente: Autores

3.6.3. DIAGRAMAS DE EVENTOS

En este formato se establecen los eventos que pueden ser generados por el actor y van a ser atendidos por cada Caso de Uso. Por evento se entiende la interacción que tiene un actor con la aplicación a través de la interfaz gráfica, tal como el clic de un ratón, el ingreso de un texto a un componente, el movimiento de un elemento de la interfaz, etc. Todos los eventos van numerados en orden secuencial de acuerdo a la secuencia lógica como ocurrirían en la aplicación (ciclo de vida del caso de uso).

Tabla 3.8: Diagrama de Eventos Autenticación

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso	Autenticación
Actor Responsable	Usuario Registrado, Administrador
EVENTO	**RESPUESTA DEL SISTEMA**
1. Ingresar al Sistema **2.** Clic en botón Registrar **4.** Ingreso de ID de usuario **5.** Ingreso de Contraseña **6.** Clic Botón Ingresar **11.a** El usuario puede ingresar al	**3.** Capturar la Selección **7.** Captura la Selección y prepara el sistema para ingresar datos **8.** Hace las respectivas validaciones. **9.** Recolecta información de la base de datos. **a. INFORMACIÓN OK** **10. a** Prepara la interfaz gráfica para el tipo de usuario ingresado.

sistema con sus respectivos privilegios si sus datos han sido bien ingresados **11. b.** Es necesario que el usuario digite nuevamente su información para volver a repetir el proceso de ingreso.	**b. INFORMACIÓN ERRADA** **10. b** El sistema emite un mensaje de datos incorrectos y no permite pasar de ese nivel de privilegio.

Fuente: Autores

Tabla 3.9: Diagrama de Eventos juegos

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso:	Juegos
Actor Responsable:	Usuario Registrado, Administrador
EVENTO	**RESPUESTA DEL SISTEMA**
1. Selecciona abrir la pantalla Jugar. **3.** Clic en botón Aceptar. **6.** El usuario Responde el cuestionario. **8.** Clic Finalizar Prueba	**2.** El sistema abre la pantalla “Aprobar Prueba” **4.** Capturar la Selección **5.** Muestra una pequeña prueba de conocimientos en Educación Vial. **7.** Capturar la Selección **9.** Capturar la Selección y prepara el sistema para validación de datos.

	a. INFORMACIÓN OK
	10. a Permite ingresar a escoger uno de los juegos proporcionados en WEB COVIAL 1.0.
11. a El usuario escoge el juego.	
	12. a Captura la selección y prepara el Sistema.
13.a El usuario inicia el juego	
	b. INFORMACIÓN ERRADA
	10.b El sistema no muestra el juego
	11.b El sistema pide al usuario que lo vuelva a intentar en otra ocasión
12.b Clic Aceptar	

Fuente: Autores

Tabla 3.10: Diagrama de Eventos Foro

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso:	Foro
Actor Responsable:	Usuario Registrado, Administrador, Usuario NO registrado.
EVENTO	**RESPUESTA DEL SISTEMA**
1. EVENTO AUTENTICACIÓN **2.** Selecciona abrir la pantalla Foro.	**3.** El sistema abre la pantalla correspondiente a Foro.

4. Clic en botón Leer Respuestas.	
	5. Capturar la Selección
	6. Mostrar los Respuestas existentes en la base de datos a una Categoría.
7. Clic en el botón Insertar.	
	8. Capturar la Selección
9. Ingreso de Datos Categoría	
	10. Validación de Datos.
	a. INFORMACIÓN OK
	11. a Guarda los Datos de Categoría.
12. a Clic en el botón Comentar/ Responder.	
	13. a Captura la selección y prepara el Sistema para la selección.
	14. a Guarda los Datos.
15. a Clic en el botón Eliminar.	
	16. a Captura la Selección
	17. a Guarda los Datos.
	b. INFORMACIÓN ERRADA
	11.b El sistema no realiza la Eliminación
	12.b El sistema pide al usuario que lo

13.b Clic Aceptar	vuelva a intentar en otra ocasión

Fuente: Autores

Tabla 3.11: Diagrama de Eventos Catálogo

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso:	Catálogo
Actor Responsable:	Usuario Registrado, Administrador
EVENTO	**RESPUESTA DEL SISTEMA**
1. EVENTO AUTENTICACIÓN **2.** Selecciona abrir la pantalla Catálogo. **4.** Clic para modificar Catálogo. **5.** El cliente escoge orden y los archivos multimedia preferidos por el usuario. **6.** El cliente guarda su modificación	**3.** El sistema abre la pantalla correspondiente a Catálogo.

Fuente: Autores

Tabla 3.12: Diagrama de Eventos Noticias

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso:	Noticias
Actor Responsable:	Usuario Registrado, Administrador
EVENTO	**RESPUESTA DEL SISTEMA**
1. EVENTO AUTENTICACIÓN **2.** Selecciona abrir la pantalla	

Noticias.	
	3. El sistema abre la pantalla correspondiente a Noticias.
4. Clic para visualizar Noticia.	
	5. Capturar la Selección
	6. Mostrar Noticia
7. Clic en el botón Crear Noticia.	
	8. Validación de Datos.
	a. INFORMACIÓN OK
	9. a Capturar la Selección
	10. a Guarda los Datos.
11. a Clic en el botón Modificar.	
	12. a Captura la selección y prepara el Sistema para modificar datos.
	13. a Guarda los Datos.
14. a Clic en el botón Eliminar.	
	15. a Captura la selección y prepara el Sistema para eliminar datos.
	16. a Guarda los Datos.
	b. INFORMACIÓN ERRADA
	9.b El sistema no realiza la Eliminación

11. b Clic Aceptar.	**10.b** El sistema pide al usuario que lo vuelva a intentar en otra ocasión

Fuente: Autores

Tabla 3.13: Diagrama de Eventos Chat

CONTROL DE PROYECTOS	FORMATO DE EVENTOS
Nombre del Caso de Uso:	Chat
Actor Responsable:	Usuario Registrado, Administrador
EVENTO	**RESPUESTA DEL SISTEMA**
1. EVENTO AUTENTICACIÓN	
2. Selecciona abrir la pantalla Chat.	
	3. El sistema abre la pantalla correspondiente a Chat.
4. Escribe Mensaje.	**a. INFORMACIÓN OK**
	5.a. Capturar la Selección
	b. INFORMACIÓN OK
7.b. Clic Aceptar	**6.b.** El sistema pide al usuario que lo vuelva a intentar en otra ocasión

Fuente: Autores

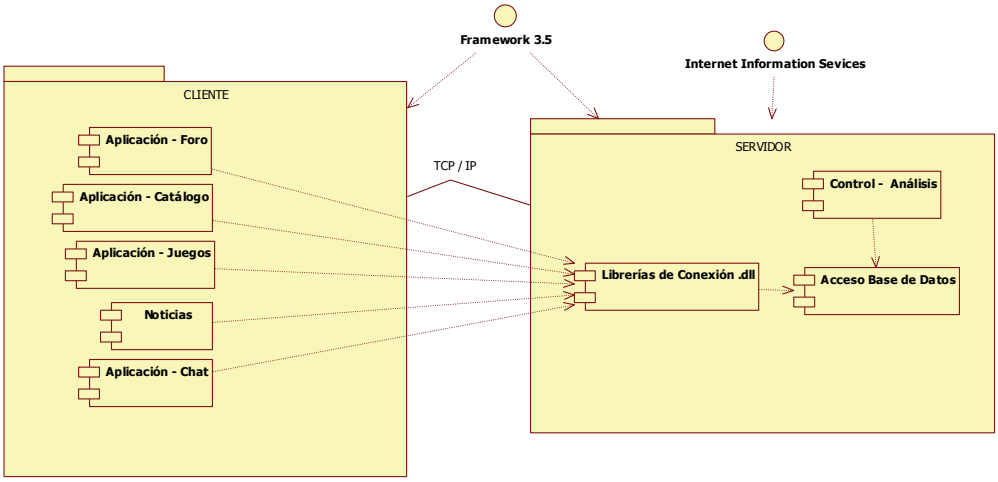
3.6.4. DIAGRAMA DE COMPONENTES

Lo que distingue el Diagrama de Componentes de otro tipo de diagramas es sin duda su contenido. Normalmente contiene componentes, interfaces y relaciones entre ellos.

Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.

Los componentes pertenecen a un mundo físico, es decir, representan a un bloque de construcción al modelar aspectos físicos de un sistema.

Figura 3.15: Diagrama de Componentes WEB COVIAL 1.0.



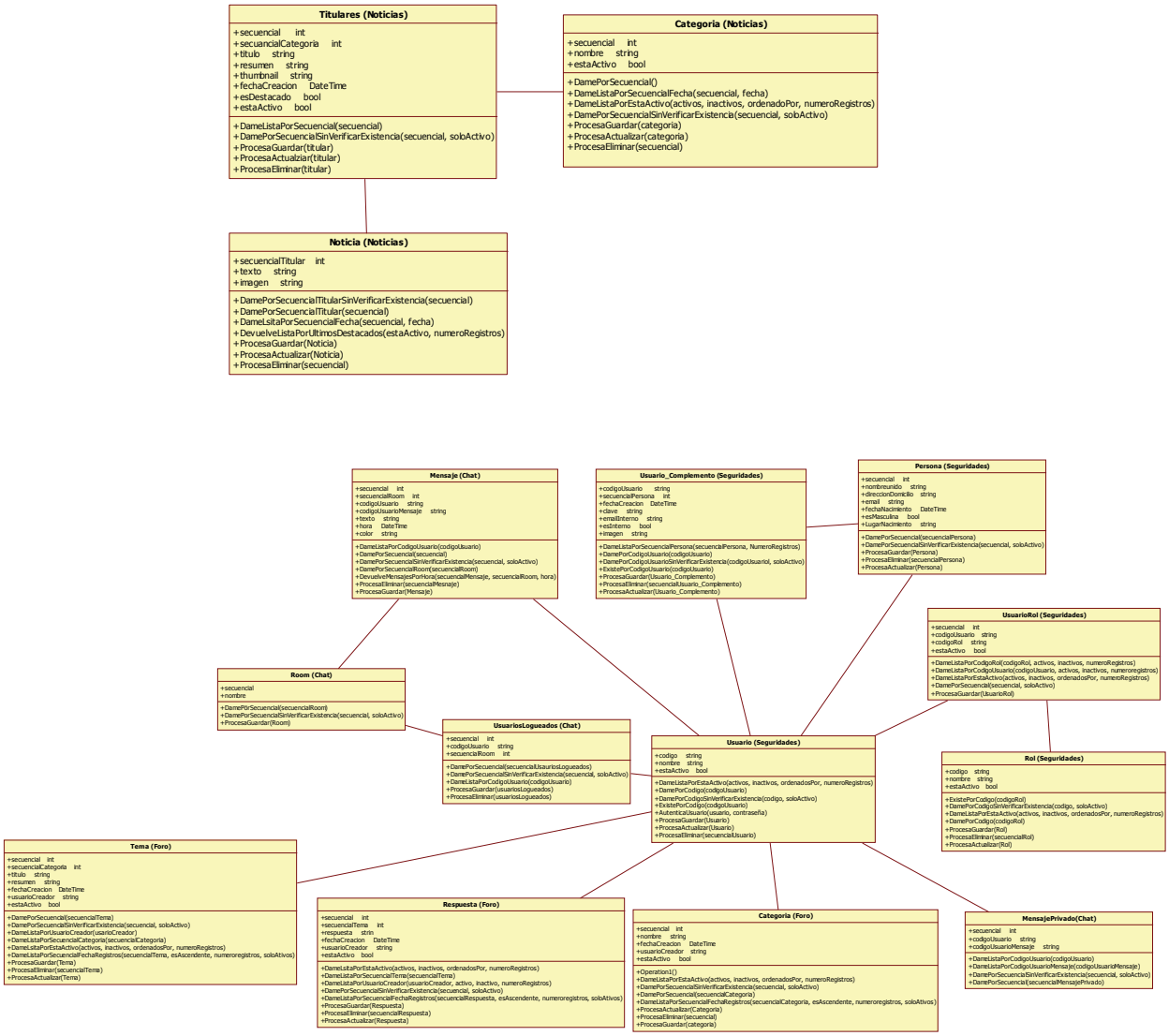
Fuente: Autores

3.6.5. DIAGRAMAS DE CLASES

Una clase define el ámbito de definición de un conjunto de objetos, por tanto cada objeto pertenece a una clase.

El Diagrama de Clases es el diagrama principal para el diseño del sistema, en el que presenta las clases con sus respectivas relaciones estructurales y de herencia. La definición de clases incluye definiciones para atributos y operaciones.

Figura 3.16: Diagrama de Clases WEB COVIAL 1.0.

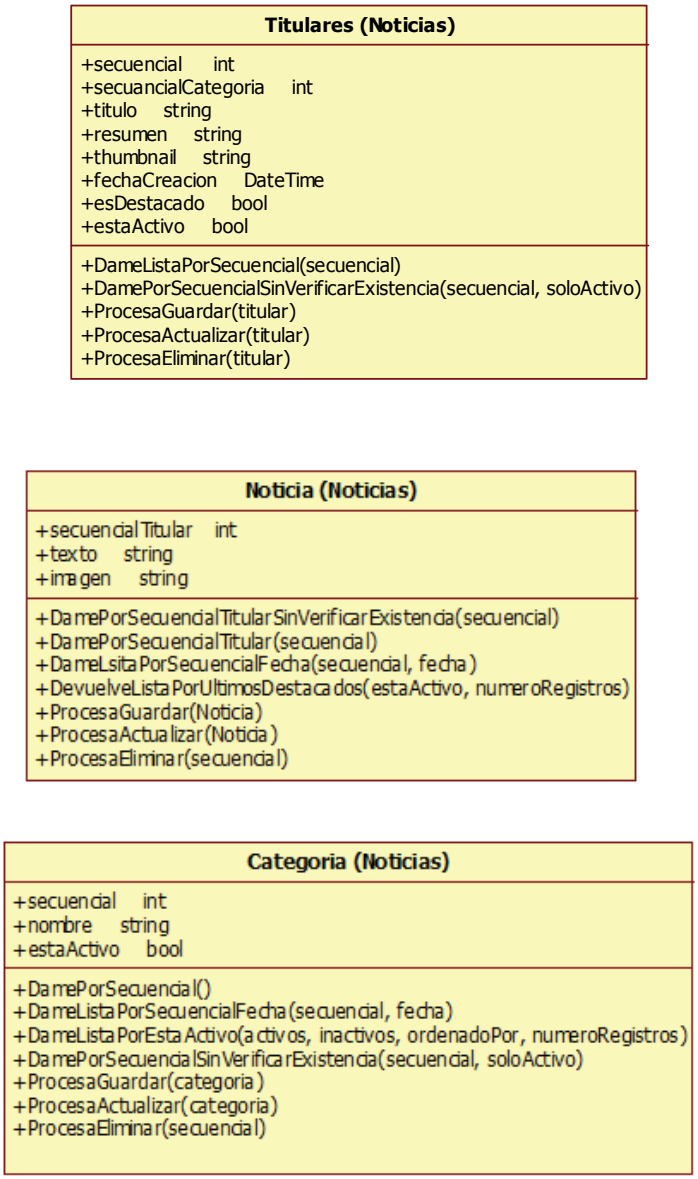


Fuente: Autores

Puesto que el diagrama de clases es muy extenso se dividirá en pequeñas secciones para que puedan ser especificadas todas.

3.6.5.1. NAMESPACE NOTICIAS

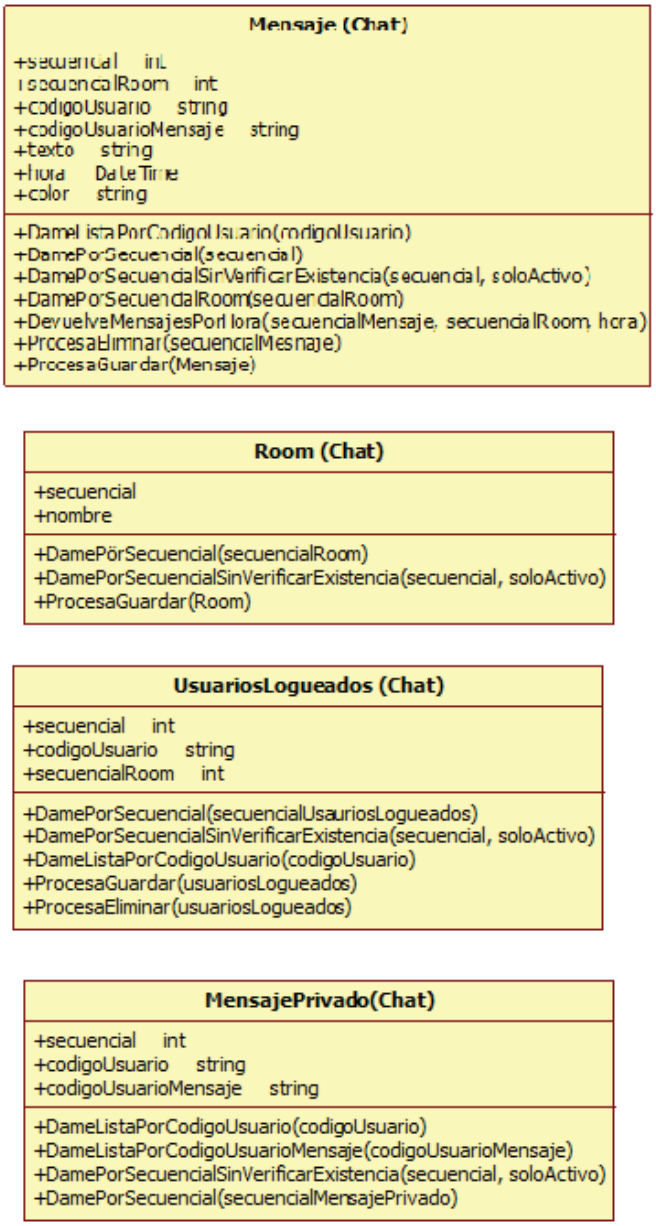
Figura 3.17: Diagrama de Clases Esquema (Noticias) WEB COVIAL 1.0.



Fuente: Autores

3.6.5.2. NAMESPACE CHAT

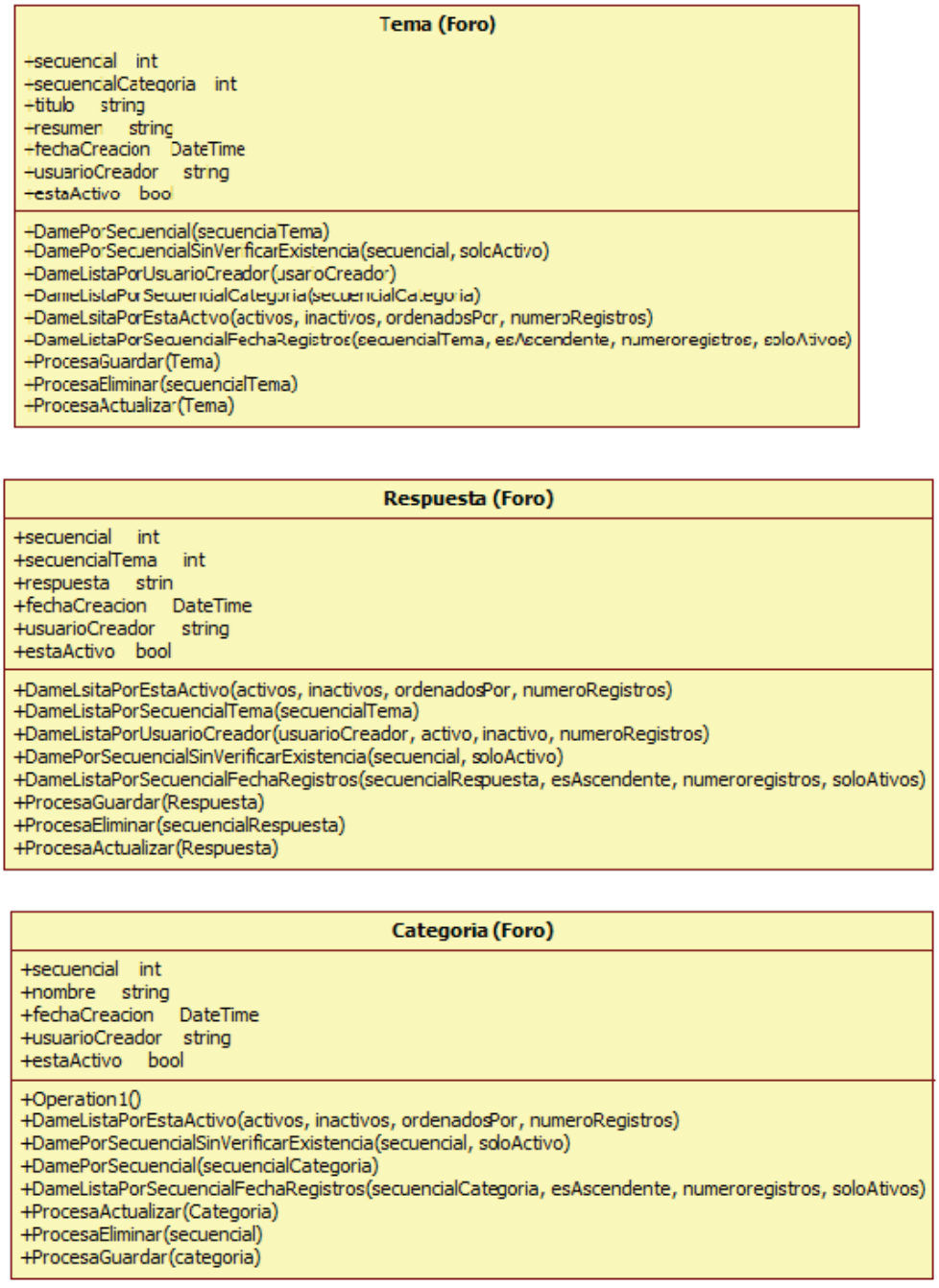
Figura 3.18: Diagrama de Clases Esquema (Chat) WEB COVIAL 1.0.



Fuente: Autores

3.6.5.3. NAMESPACE FORO

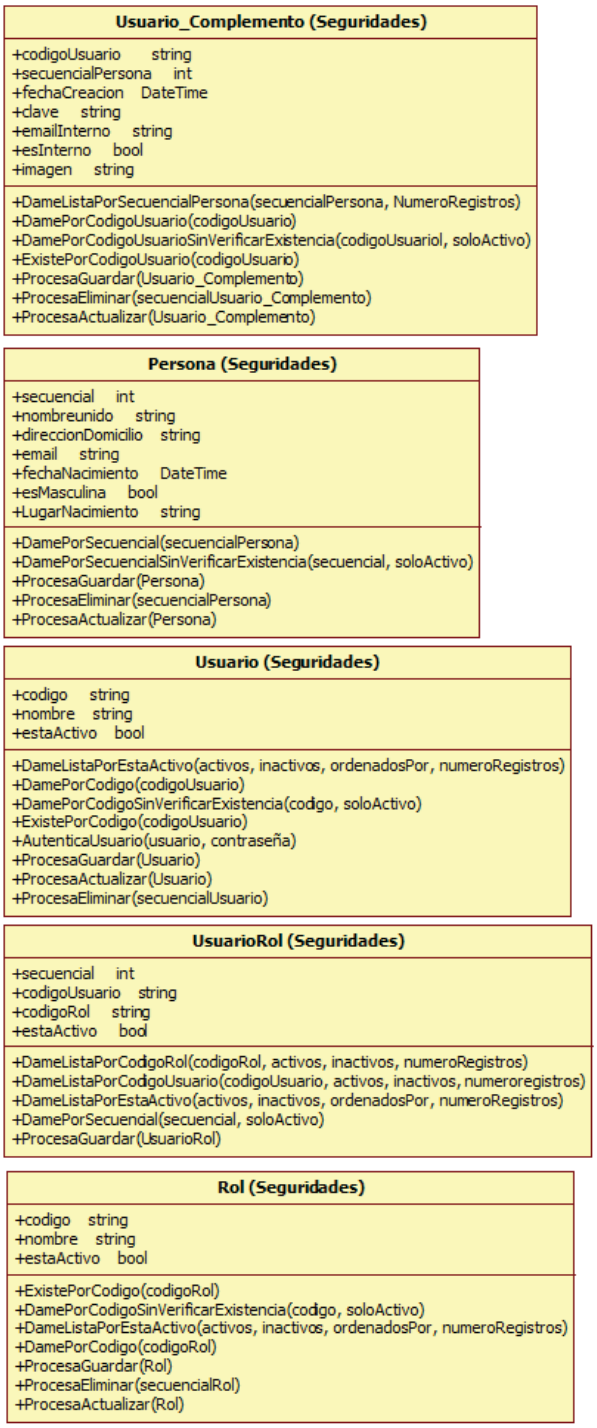
Figura 3.19: Diagrama de Clases Esquema (Foro) WEB COVIAL 1.0.



Fuente: Autores

3.6.5.4. NAMESPACE SEGURIDADES

Figura 3.20: Diagrama de Clases Esquema (Seguridades) WEB COVIAL



Fuente: Autores

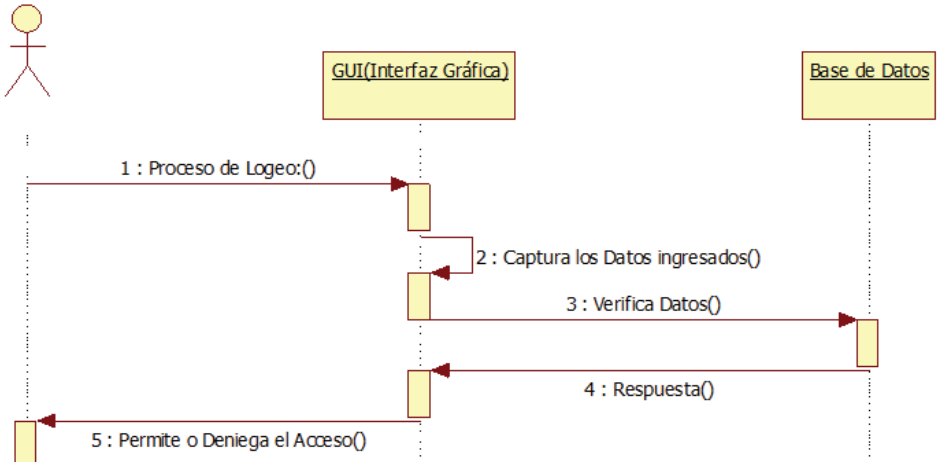
3.6.6. DIAGRAMAS DE ITERACIONES

Este diagrama muestra las interacciones de un usuario con el sistema. Una Interacción es una cadena de mensajes enviados en respuesta a un evento generado por el usuario sobre la aplicación.

El responsable o ACTOR es quien inicia el ciclo interactuando inicialmente con la interfaz de usuario: GUI; enseguida se inician todos los objetos que intervienen en el funcionamiento del aplicativo. En este diagrama se comienza a observar el comportamiento del sistema a partir de los eventos generados por los actores.

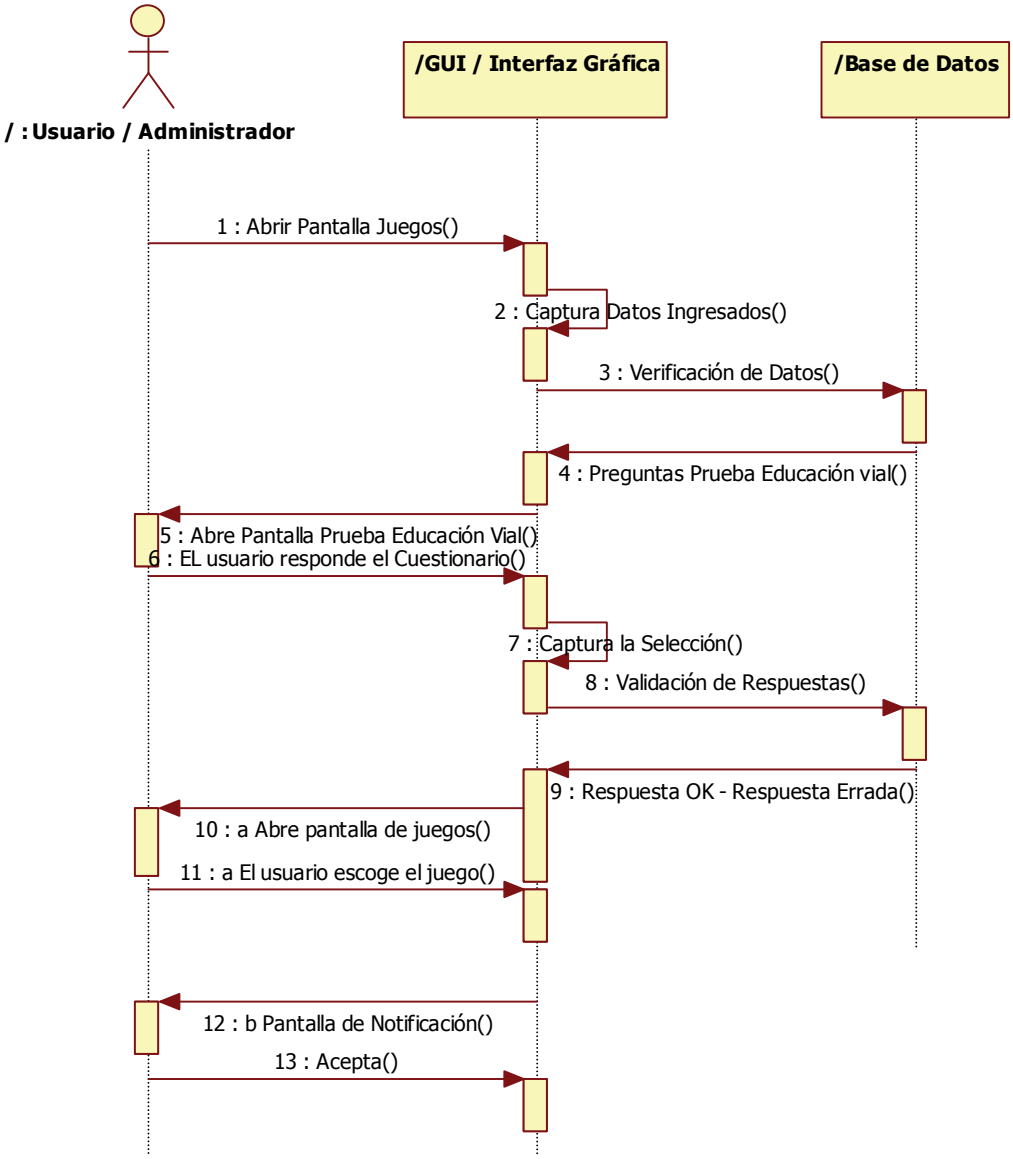
De igual manera se realizará solamente un ejemplo (Autenticación):

Figura 3.21: Diagrama de Iteraciones – Sub-Caso de Uso Autenticación



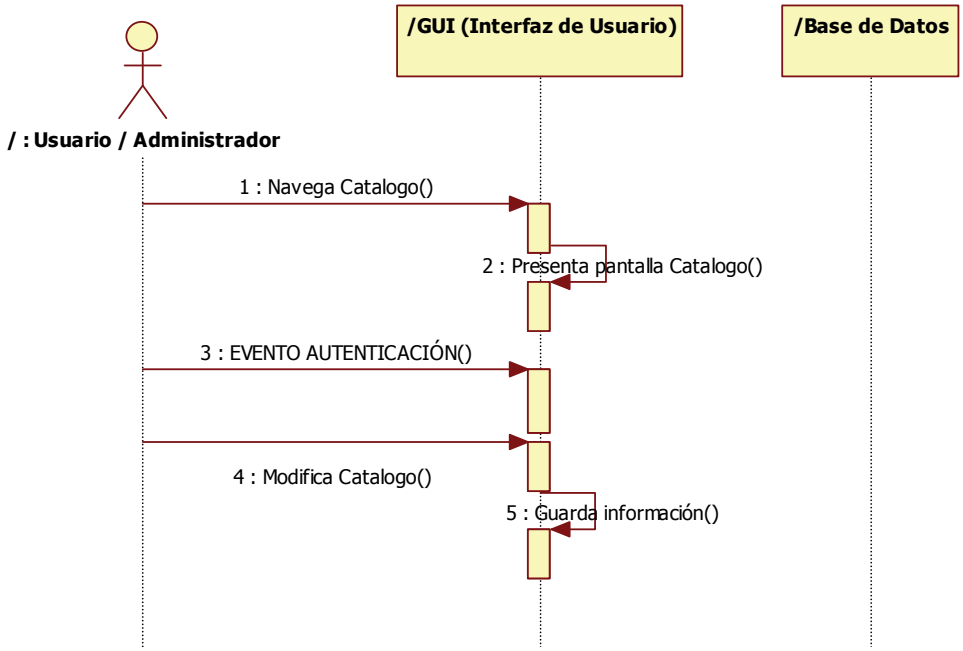
Fuente: Autores

Figura 3.22: Diagrama de Iteraciones – Sub-Caso de Uso Juegos



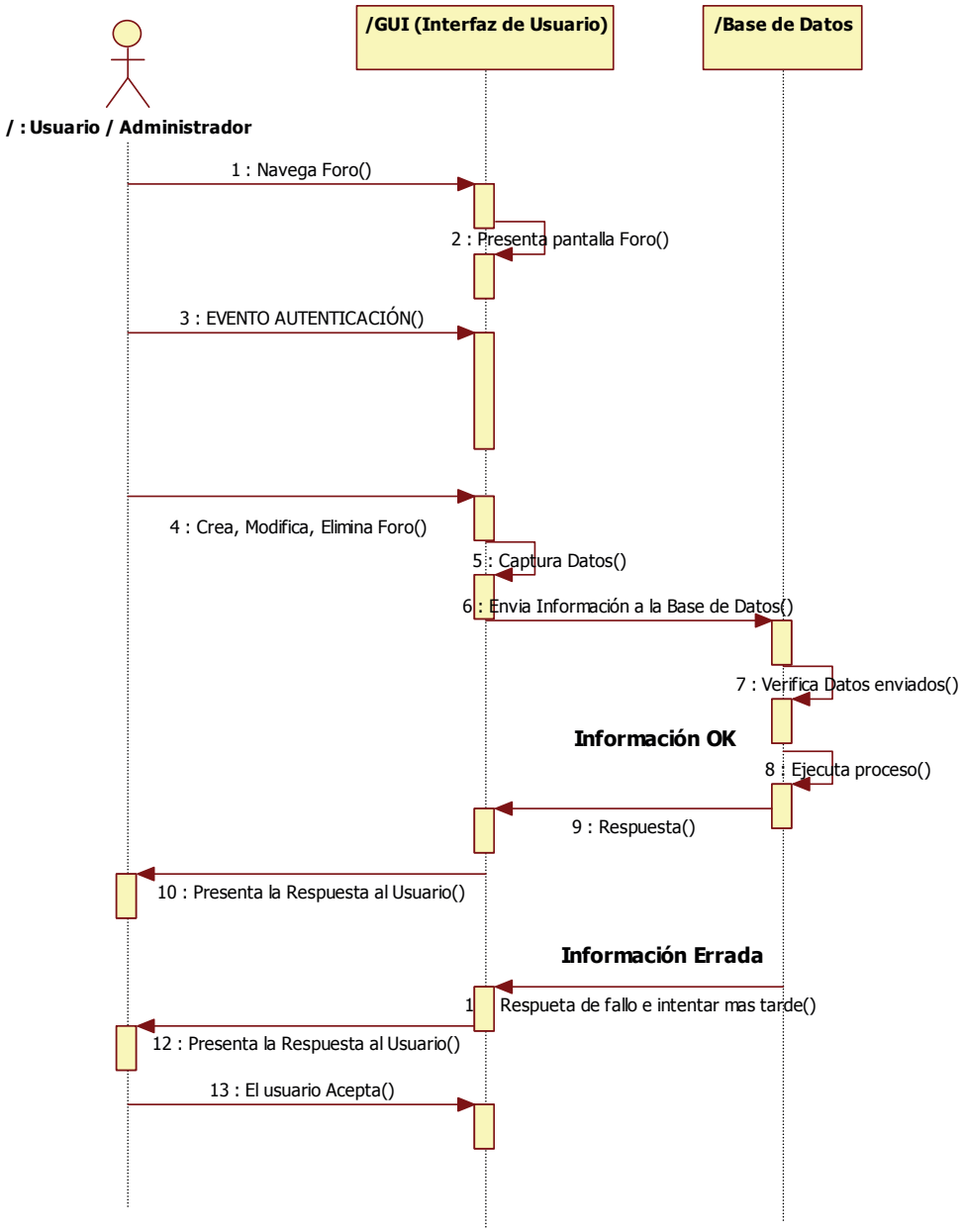
Fuente: Autores

Figura 3.23: Diagrama de Iteraciones – Sub-Caso de Uso Catálogo



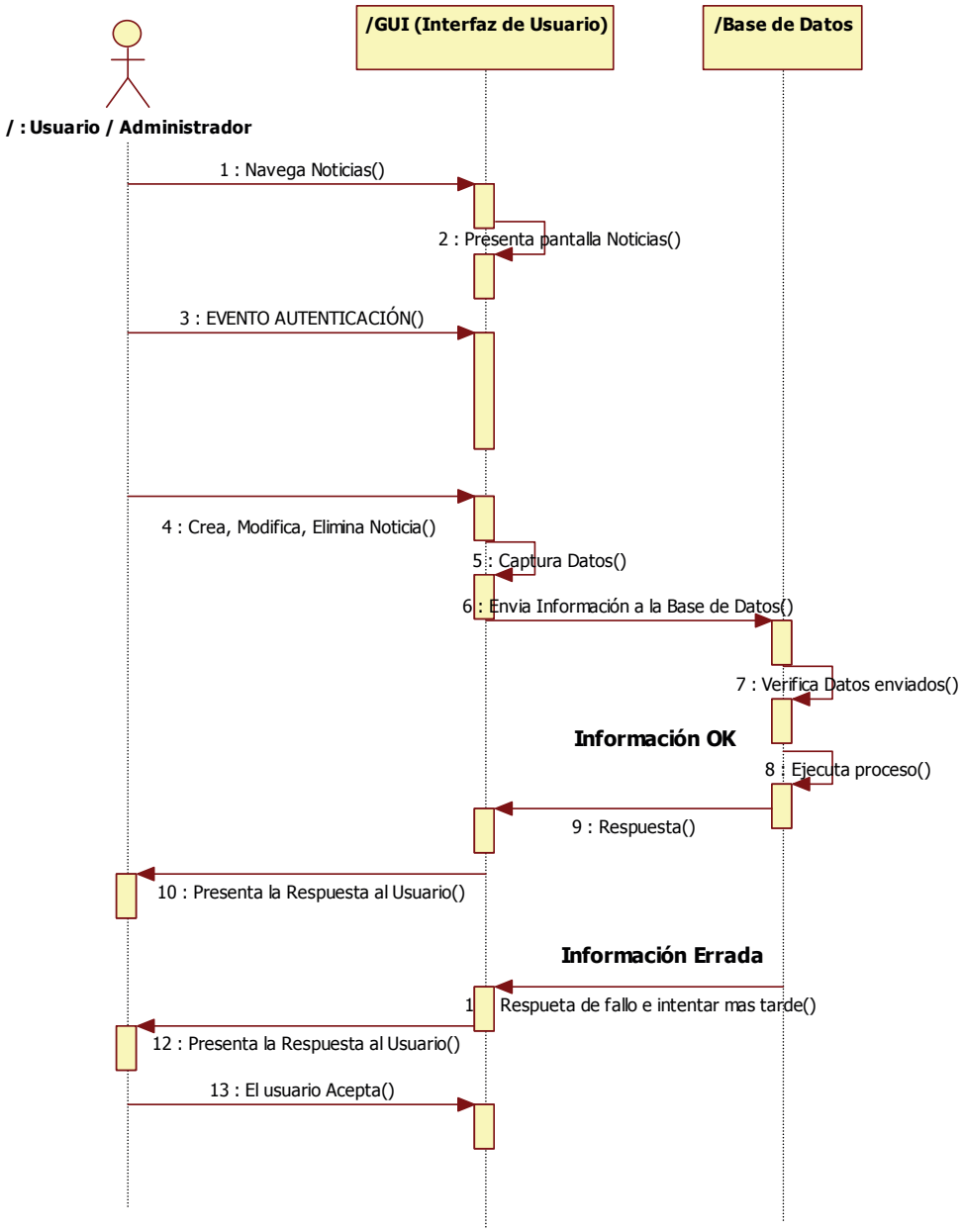
Fuente: Autores

Figura 3.24: Diagrama de Iteraciones – Sub-Caso de Uso Foro



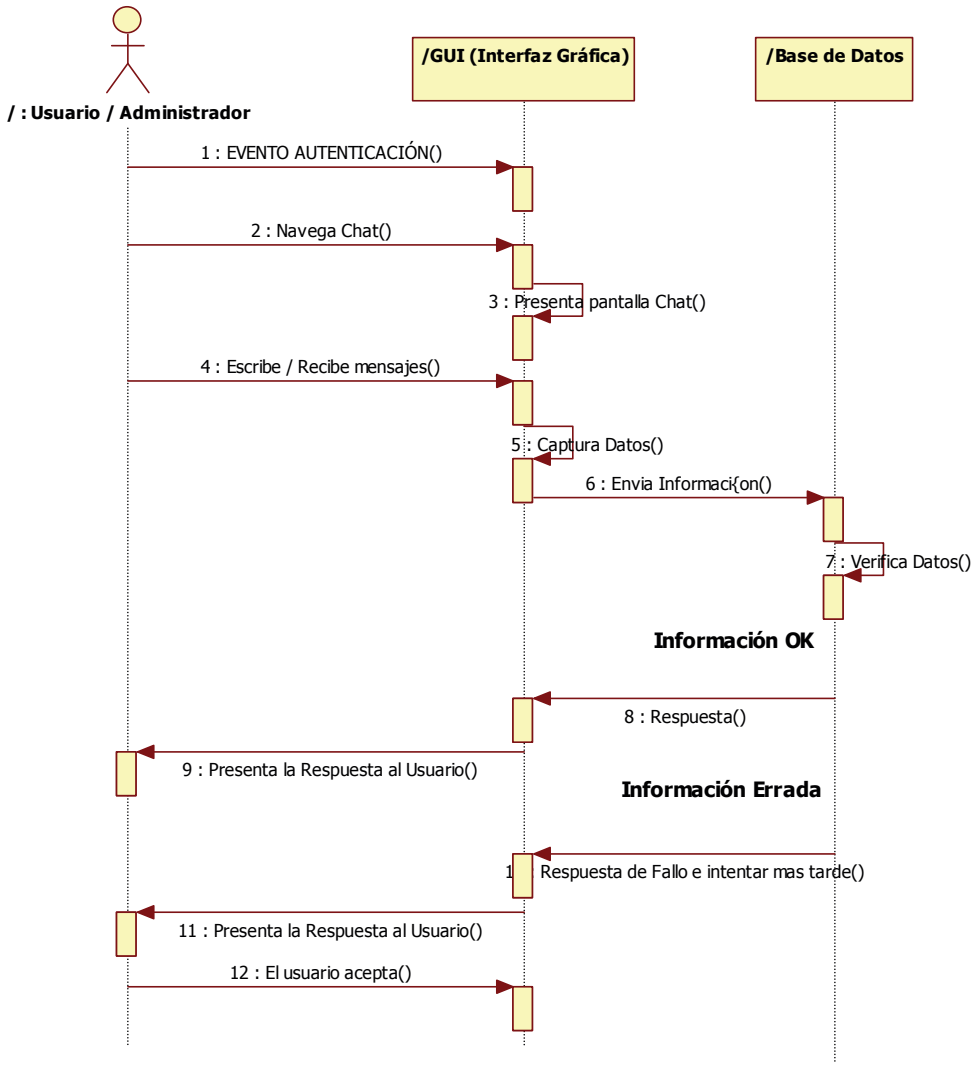
Fuente: Autores

Figura 3.25: Diagrama de Iteraciones – Sub-Caso de Uso Noticias



Fuente: Autores

Figura 3.26: Diagrama de Iteraciones – Sub-Caso de Uso Chat



Fuente: Autores

3.7. DESARROLLO DE LA APLICACIÓN.

COVIAL WEB 1.0, se encuentra definido mediante tres proyectos básicos:

Tabla 3.14: Descripción de Proyectos WEB COVIAL 1.0.

PROYECTO	herramienta de desarrollo	Lenguaje de programación	Descripción
SGBD	Sql Server 2008	Sql	Desarrollo de una Base de Datos Relacional COVIAL
COVIAL SERVIDOR	Visual Studio 2008	C#,	Definición de Clases, Objetos, Métodos y Servicios
COVIAL CLIENTE	Visual Studio 2008	C#, XAML	Desarrollo Visual de la Aplicación al Usuario.

Fuente: Autores

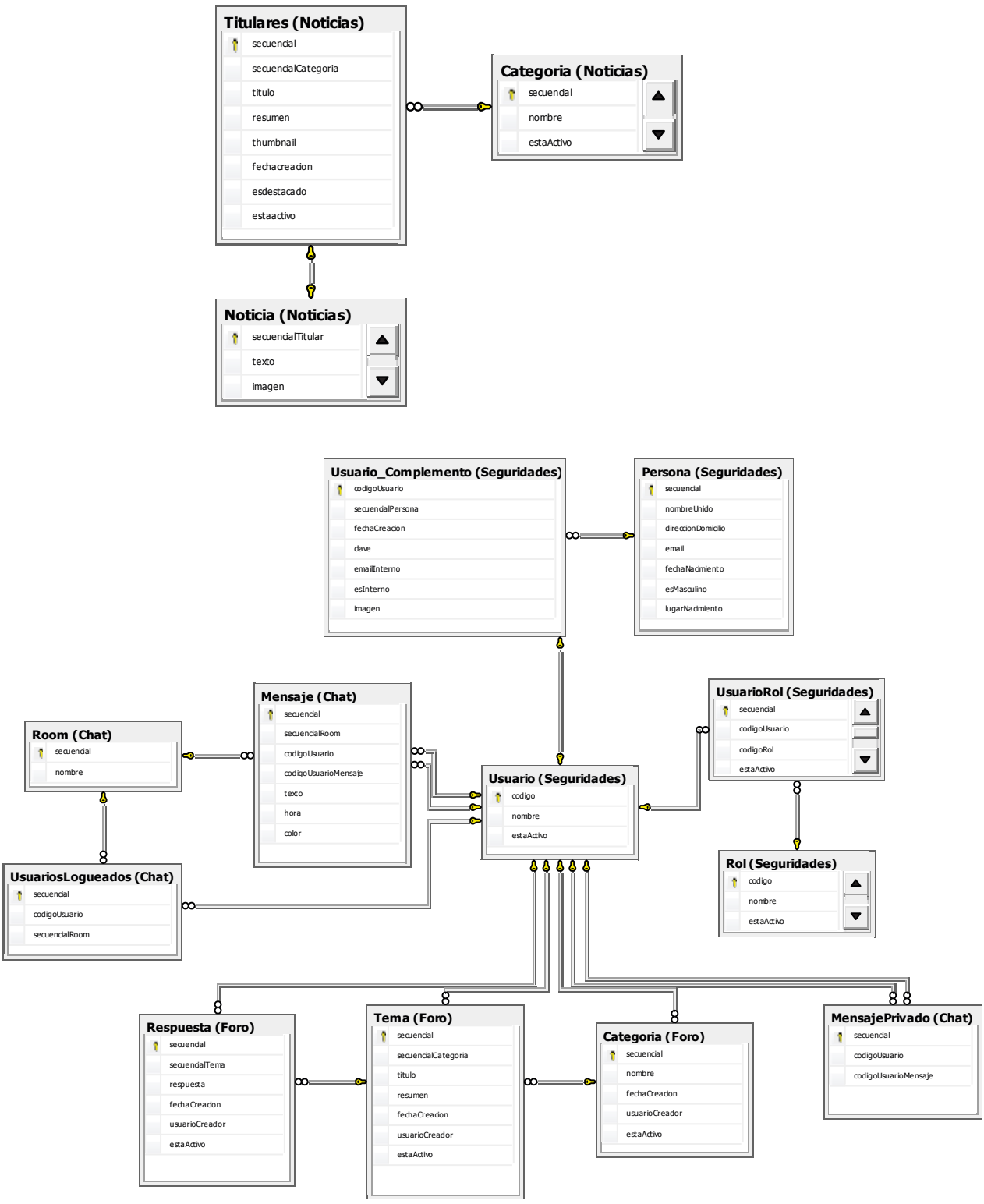
3.7.1. SGBD EN WEB COVIAL 1.0

Como se puede ver en la **figura 3.16,** La Base de Datos de WEB COVIAL 1.0 se encuentra diseñada por esquemas bien definidos (Chat, Noticias, Seguridades, Foros); cada uno destinados a dar una funcionalidad al Sistema.

Se puede observar que la Clase base del esquema SEGURIDADES es Usuario con la cual la mayoría de tablas se relacionan.

Vale la pena aclarar que se ha aprovechado las ventajas de SQL para garantizar integridad referencial mediante claves primarias y foráneas.

Figura 3.27: Diagrama de Base de Datos WEB COVIAL 1.0



Fuente: Autores

3.7.2. ESTANDARIZACIÓN WEBCOVIAL 1.0

3.7.2.1. SERVIDOR

3.7.2.1.1. Convenciones de Nombres y Estilos

- Se omitirán las tildes en todos los casos.
- Las eñes se reemplazaran por la silaba “ni”.
- No se utilizarán abreviaturas de ningún tipo, en todos los casos se utilizarán nombres completos sin ninguna separación en caso de que estén formados por varias palabras.
- Todo elemento expuesto en la parte inferior deberá tener antes de su definición, una descripción de su naturaleza, de manera que esta descripción sea utilizada para la generación de documentación XML.

Ejemplo:

```
/// <summary>
/// Propiedad que indica la Hora de Ingreso
/// </summary>
```

3.7.2.1.1.1. Atributos

- Comience con un guión bajo (“_”)
- Use convenciones de nombres tipo CAMEL

  Ejemplo:

  _codigo, _estaActivo

3.7.2.1.1.2. Variables y Parámetros

- Use convenciones de nombres tipo CAMEL
  Ejemplo:
  contador, indice, primerRegistro

3.7.2.1.1.3. Propiedades

- Use convenciones de nombres tipo PASCAL
  Ejemplo:
  Codigo, EstaActivo

3.7.2.1.1.4. Métodos y Funciones

- Use convenciones de nombres tipo PASCAL
  Ejemplo:
  CambiaClaveEnLote, ProcesaGuardar

3.7.2.1.1.5. Nombres de Clases

- Utilice nombres en singular
- Use convenciones de nombres tipo PASCAL

Ejemplo:

Usuario, EmpresaParametro

3.7.2.1.2. Tipos de Clases

- Todas las clases estarán relacionadas directamente con una tabla en la base de datos.

- Todos los namespaces corresponderán a un proyecto de clases ubicados en la carpeta Core.
- El namespace en donde se ubique una clase tendrá el mismo nombre que el esquema en el que la tabla se encuentra.

Ejemplo:

```
Namespace Seguridades
```

3.7.2.1.2.1. Entidad

- Representa la estructura de una tabla de la base de datos, y su nombre de clase será el mismo que el de la tabla. Su nombre de archivo estará formado por el nombre de la tabla seguido de la expresión _Entidad.
- Toda entidad heredará de la clase denominada: ClaseBase, la misma que tiene la información de la existencia de un objeto en la base de datos por medio del campo denominado existeEnRepositorio.

Es de tipo parcial y su acceso será `público`.

Ejemplo:

Nombre de archivo: Usuario_Entidad.cs

Clase: `publica partial class Usuario:ClaseBase`

- En caso de que se requiera adicionar atributos, propiedades y métodos con funcionalidad adicional a la expuesta por la tabla en la base de datos se deberá crear un nuevo archivo con el nombre de la entidad, seguida de la expresión _EntidadNegocio, que representará una extensión de la clase parcial.

Ejemplo:

Nombre de archivo: Usuario_EntidadNegocio.cs

Clase: `public partial class Usuario`

- Está formada por las siguientes regiones de código:

a) Atributos.- representarán cada uno de los campos de la tabla en la base de datos, más el atributo llamado _numeroVerificadorOriginal en caso de que la entidad tenga numeroVerificador, todo atributo es privado.

b) Propiedades.- que expondrán los atributos de manera pública, se deberán hacer diferenciaciones entre los atributos editables y los no editables.

c) Información de la Base de Datos.- Contendrá propiedades públicas de tipo estático que contienen el nombre de la tabla, los nombres de todos los campos y los tipos de datos que les corresponden, estas propiedades serán utilizadas para la generación de comandos SQL.

Ejemplo:

```
Public static string NombreTabla

{
get{ return "Seguridades.Usuario"; }
}
Public static string CodigoCampo
{
get{ return "Usuario.codigo"; }
}
public static System.Data.DbType CodigoTipo
{
get{return System.Data.DbType.AnsiString; }
}
```

d)Constructores.- Toda entidad tendrá 3 constructores; el primero tendrá como parámetro una entidad del mismo tipo y será de tipo `internal`; el segundo tendrá como parámetros todos los campos de la base de datos incluido el `numeroVerificador` y el campo

`existeEnRepositorio` y será de tipo `internal`, el primer y segundo constructores en su interior hacen una instanciación de todos los atributos de la clase; el tercer es un constructor de tipo `public` y tendrá como parámetros todos los campos de la base de datos sin incluir los campos identidad (auto-generables) en caso de que existan, el `numeroVerificador` y el campo `existeEnRepositorio`, este constructor implementará una llamada al segundo constructor para su instanciación, indicando como `numeroVerificador` el valor 0 (cero) y como `existeEnRepositorio` el valor false.

e) **Métodos.-** región que permite disponer de funciones que realizarán tareas específicas sobre la entidad.

En todos los casos se considera necesaria la función: `ActualizaRepositorio`, la misma que sirve para actualizar el atributo de persistencia `_existeEnRepositorio` en caso de que exista una inserción de un registro nuevo;
En caso de que la entidad sea editable se consideran necesarias las funciones: `ActualizaValoresOriginales`, la misma que sirve para actualizar el número verificador original en caso de que exista una edición del registro; `IncrementaNumeroVerificador`, la misma que sirve para incrementar el número verificador de concurrencia en cada actualización del registro;

En caso de que la entidad tenga un campo identidad (auto-generable) se considera necesaria la función:

`ActualizaSecuencial`, la misma que sirve para actualizar el campo identidad una vez insertado el registro.

- En caso de que uno de los atributos corresponda a un campo en otra entidad, esto es que la tabla tiene una relación, deberá implementarse de la siguiente manera:

    - Se incluirá al final de la sección de atributos un espacio para los atributos foráneos, en donde se ubicará su descripción.

    - A continuación se incluirá un espacio para los atributos privados de entidades relacionadas, para identificar los objetos relacionados por los atributos foráneos

    Ejemplo:

```
Private int _secuencialOficinaEmpresa;
PrivateOficinaEmpresa_oficinaEmpresa;
```

3.7.2.1.2.2. DALC (Data Access Layer Component o Componente de Capa de Acceso a Datos)

Clase de tipo `internal` que contiene un conjunto de métodos estáticos de tipo `internal` que efectuarán operaciones de tipos CRUD sobre una tabla en la base de datos.

- Su nombre de archivo estará formado por el nombre de la tabla seguido de la expresión _DALC

Ejemplo:
Nombre de archivo: Usuario_DALC.cs
Clase: `internal class UsuarioDALC`

De acuerdo a los datos de la tabla se sugiere la siguiente estructura en este orden:

a)Método `ExistePor[ClavePrimaria](tipo campo)`, que permite verificar la existencia de un registro cuya clave primaria será igual a campo. El método aparece si la clave primaria no es un secuencial.

b)Métodos `ExistePor[Campo1Campo2..](tipo1campo1,tipo2campo2,..)`, que permite verificar la existencia de un registro cuyos datos correspondan a un índice único compuesto por los campos indicados como parámetros.

c)Método `DamePor[ClavePrimaria](tipo campo)`, que permite obtener un único objeto cuya clave primaria será igual a campo.

d)Métodos `DamePor[Campo1Campo2..](tipo1campo1, tipo2campo2,..)`, que permite obtener un único objeto cuyos datos correspondan a un índice único compuesto por los campos indicados como parámetros.

e)Método `DameUno(ObjetoCondicionobjetoCondicion)`, que permite obtener un objeto de acuerdo a los parámetros expuestos en el objeto Condición.

f)Método `DameLista(ObjetoCondicion objetoCondicion)`, que permite obtener una lista de objetos de acuerdo a los parámetros expuestos en el objetoCondicion.

g)Método `Guardar(Objeto objeto)`, que permite insertar o editar un registro.

h)Método `Borrar(Objeto objeto)`, que permite borrar un registro.

i) **Método** `CreaComandoInsert(Database db, Objeto objeto)`, que permite obtener un comando del tipo `DbCommand`, que se utilizará para insertar un registro con los datos del objeto.

j) **Método** `CreaComandoSelect ()`, que permite obtener una cadena, que se utilizará para leer registros, la cadena será del tipo "`SELECT nombreTabla.nombreCampo1, nombreTabla.nombreCampo2,.. FROM nombreTabla`".

k) **Método** `CreaComandoUpdate(Database db, Objeto objeto)`, que permite obtener un comando del tipo `DbCommand`, que se utilizará para actualizar un registro con los datos del objeto.

l) **Método** `CreaComandoDelete(Database db, Objeto objeto)`, que permite obtener un comando del tipo `DbCommand`, que se utilizará para borrar un registro con los datos del objeto.

3.7.2.1.2.3. Actor

- Clase de tipo `public partial` que contiene un conjunto de métodos estáticos de tipo `public` que efectuarán operaciones sobre los objetos utilizando los métodos expuestos en las clases de tipo DALC.
- Su nombre de archivo estará formado por el nombre de la tabla seguido de la expresión _Actor
- Su nombre de clase estará formado por el nombre de la tabla seguido de la expresión Actor

Ejemplo:

Nombre de archivo: Usuario_Actor.cs

Clase: `public partial class UsuarioActor`

De acuerdo a los datos del objeto se sugiere la siguiente estructura en este orden:

a) Método `ExistePor[ClavePrimaria](tipo campo)`, que hace uso del método correspondiente en la clase DALC para verificar la existencia de un objeto.

b)Métodos `ExistePor[Campo1Campo2..](tipo1campo1, tipo2campo2,..)`, que hace uso del método correspondiente en la clase DALC para verificar la existencia de un objeto.

c) Método `DamePor[ClavePrimaria](tipo campo)`, que hace uso del método correspondiente en la clase DALC para obtener un objeto.

d) Métodos `DamePor[Campo1Campo2..](tipo1campo1, tipo2campo2,..)`, que hace uso del método correspondiente en la clase DALC para obtener un objeto.

e) Método `DameUno(ObjetoCondicion objetoCondicion)` que hace uso del método correspondiente en la clase DALC para obtener un objeto.

f) Método `DameLista(ObjetoCondicion objetoCondicion)`, que hace uso del método correspondiente en la clase DALC para obtener una lista de objetos.

3.7.2.1.2.4. Actor Negocio

- Clase de tipo `public partia`l que contiene un conjunto de métodos estáticos de tipo `public` que efectuarán operaciones diversas sobre los objetos.
- Su nombre de archivo estará formado por el nombre de la tabla seguido de la expresión _ActorNegocio.

- Su nombre de clase estará formado por el nombre de la tabla seguido de la expresión ActorNegocio.

Ejemplo:

Nombre de archivo: Usuario_ActorNegocio.cs

Clase: `public partial class UsuarioActorNegocio`

De acuerdo a la naturaleza de las operaciones se sugiere la siguiente estructura en este orden:

a) Métodos de lectura de registro único (`DamePor..`).

b) Métodos de lectura de varios registros (`DameListaPor..`).

c) Métodos de inserción/actualización de registros (`ProcesaGuardar, ProcesaNuevo, ProcesaActualizar`).

d) Método de borrado de registro (`ProcesaBorrar`).

e) Métodos de Negocio.

3.7.2.1.2.5. Mensajería

- Las clases de mensajería corresponden a los objetos que viajarán entre el servidor y la aplicación cliente.
- El proyecto de clases de mensajería está nombrado como Mensajería, el mismo que estará formado por un conjunto de carpetas correspondientes a los proyectos de clases del Core.
- Cada carpeta contendrá varios archivos en los que estarán definidos los mensajes para cada Entidad que pertenezca a dicho esquema.
- El nombre del archivo será el mismo nombre de Entidad seguida de Mensajes.

Ejemplo: Nombre de archivo: `UsuarioMensajes.cs`

- Cada archivo contendrá varias clases relacionadas con la entidad. Las clases serán creadas con acceso `public`.
- De manera general los mensajes que provienen del Cliente tienen la extensión `ME` y los que salen del Servidor tienen la extensión `MS`, sin que esta característica sea obligatoria para todos los mensajes.
- Los mensajes se han divido en 3 grupos en orden de prioridad:
- **Mensajes de Entidad**, Para los mensajes que respondan a los datos de una Entidad en concreto. Se ubicará un archivo por cada Entidad en la carpeta de nombre del proyecto al que corresponde en el Core, dentro de este archivo se ubicarán un grupo de clases que corresponderán a mensajes de los siguientes tipos en orden:

  a) **ítem** de la entidad que contendrá la estructura necesaria para ser usado en el cliente en **ComboBox** o **ListBox** o en controles de búsqueda. Su estructura por lo general mantiene un campo que permita identificar al registro como único y un nombre que muestre la descripción del mismo. Su nombre será el de la entidad a la que representa seguida de la palabra **Item**.

  Ejemplo:

```
public class UsuarioItem
{
private string _codigo;
private string _nombre;
            ...
```

  b) **Lista** de ítems que contiene un arreglo estático de objetos de tipo **EntidadItem**. Su nombre será el nombre de la entidad ítem seguida de la expresión **Lista**.

  Ejemplo:

```
Public class UsuarioItemLista
{
private UsuarioItem[] _usuarios;
...
```

c) Entidad **Resumen** que contiene un mensaje con los atributos más relevantes de una entidad para ser mostrados en una lista descriptiva en el cliente. Su nombre está formado por el nombre de la entidad seguida de la expresión **Resumen**.

Ejemplo:

```
Public class UsuarioResumen
{...
```

d) **Lista** de entidades resumen que contiene un mensaje con un arreglo estático de entidades **resumen**. Su nombre está formado por el nombre de la entidad resumen seguida de la expresión **Lista**.

Ejemplo:

```
Public class UsuarioResumenLista
{
Prívate UsuarioResumen[] _Usuarios;
...
```

e) **EntidadMSE** que contiene un mensaje con todos los campos de una entidad a fin de que se pueda realizar ser mostrada en su totalidad. Su nombre está formado por el nombre de la entidad seguida de la expresión MSE.

Ejemplo:

```
Public class UsuarioMSE
{...
```

f) **Requisitos** que representará un objeto con todas las listas de ítems que junto con el mensaje **EntidadMSE** se requieren para un mantenimiento de la Entidad. Su nombre está formado por el nombre de la entidad seguida de la expresión **Requisitos**.

Ejemplo:

```
Public class UsuarioRequisitos
{...
```

g) **Mensajes Específicos,** Para los mensajes que respondan a los datos de una **Funcionalidad** en concreto, los mismos que **no**

serán reutilizados en ningún caso y **no representan a una entidad** en concreto. Se ubicarán en el archivo de la Entidad a la que corresponda el Servicio Web desde donde se lo utiliza, una característica importante es que un Mensaje Especifico siempre estará formado por uno o varios Mensajes Comunes o de clase, más datos simples que no correspondan a un atributo de ninguna entidad en el sistema. Se nombrarán con el nombre del método Web que los llama más el sufijo **MS**, **ME o MSE**, dependiendo si son mensajes de entrada, salida o ambos.

Ejemplo:

```
Public class DevuelveUsuarioPorCodigoConRequisitosMS
```

3.7.2.1.2.6. Servicios

- Los servicios son los encargados de exponer la funcionalidad que se genera en los Actores hacia el mundo exterior, es decir al cliente.
- El proyecto de clases de servicio está nombrado como **Servicios,** el mismo que estará formado por un conjunto de carpetas correspondientes a los proyectos de clases de **Core.**
- Cada carpeta contendrá varios archivos en los que estarán definidos los mensajes para cada Entidad que pertenezca a dicho archivo de proyectos de clases **Core.**
- La clase de acceso debe ser `public` y deberá ser heredada de la clase `system.web.Servicies.Webservicos` y llevar la documentación necesaria así como los atributos.

  Ejemplo:

  Nombre de Archivo: `UsuarioWS.cs`
- Cada servicio contendrá varios métodos debidamente documentados los mismos que serán los encargados de ejecutar métodos de los actores y realizar el mapeo correspondiente de datos tanto de los mensajes de entrada como los de salida de ser el caso.

  El nombre del parámetro de entrada como del objeto de retorno deberán ser **mensajeEntrada** y **mensajeSalida** respectivamente.

- Los métodos creados que se quiere publicar en la Web deben tener el acceso `public`
- De acuerdo a la naturaleza de las operaciones se sugiere la siguiente estructura en este orden:
  - **a)** Métodos de verificación de existencia `(ExistePor…)`
  - **b)** Métodos de lectura de registro único `(DevuelveUsuarioPor…)`
  - **c)** Métodos de retorno de Requisitos y Asociados de ser el caso.
  - **d)** Métodos de lectura de ítems. `(DevuelveItemLista…)`
  - **e)** Método de lectura de varios registros. `(DevuelveListaResumenPor…)`
  - **f)** Método de inserción/actualización de registros `(Nuevo, Actualiza)`
  - **g)** Método de borrados de registros `(Borra)`
  - **h)** Métodos del Negocio
- El proyecto de servicio que se expondrá hacia el exterior esta nombrado como **WebServices,** el mismo que estará formado por un conjunto de carpetas correspondientes a los proyectos de clases del Core.

3.7.2.1.2.7. Archivos de Configuración

- El archivo que contendrá todas las configuración es a nivel de la aplicación servidor es el archivo **Web.Config,** el mismo que está ubicado en el proyecto WebServices.
- Para cada tipo de configuración se definirá una sección en el archivo de configuración.

  Ejemplo:
  - **a)** Clases Bases.
  - **b)** Conexión y manipulación de Base de Datos.
  - **c)** Manejo de Excepciones propias del Sistema.

3.7.2.1.2.8. Utilidades Generales

- Se tiene una clase por cada proyecto del Core en donde se pondrán los métodos que se consideren generales para el respectivo módulo, además existe la clase Utilidades que se utiliza para cualquier método que no tenga que ser directamente con un módulo en especial.
- Todos los métodos de Utilidades deberán ser tipo de `public static`, y tener los comentarios necesarios para su uso y distribución.

3.7.2.2. CLIENTE

3.7.2.2.1. Convenciones de Nombres y Estilos

- Se omitirán las tildes en todos los casos
- Las enes se reemplazaran por la silaba “ni”.
- No se utilizarán abreviaturas de ningún tipo, en todos los casos se utilizarán nombre completos y en general sin ninguna separación en caso de que están formados por varias palabras, a menos de que la regla de nomenclatura exija un separador en concreto.

3.7.2.2.1.1. Atributos

- Comience con un guión bajo (“_”).
- Use convenciones de nombres tipo CAMEL.
  Ejemplo:
  _codigo, _estaActivo.

3.7.2.2.1.2. Variables y Parámetros

- Use convenciones de nombres tipo CAMEL.
  Ejemplo:
  contador, indice, primerRegistro.

3.7.2.2.1.3. Propiedades

- Use convenciones de nombres tipo PASCAL.
  Ejemplo:
  Codigo, EstaActivo.

3.7.2.2.1.4. Métodos y Funciones

- Use convenciones de nombres tipo PASCAL.
  Ejemplo:
  GuardarItem, RevisaNombre.

3.7.2.2.1.5. Nombres de Controles de Front End

- Usar Convenciones de nombres tipo CAMEL.
- Su nombre estará formado por el dato al que representa más el nombre del control que se muestra, en caso de que el control no tenga relación con un dato en especial y sea único en la pantalla se puede tomar el nombre del control sin ningún texto adicional, en cualquier otro caso se deberá poner un nombre descriptivo de su función más el nombre del control.

  Ejemplo:
  codigoUsuarioTextBox, secuencialPersonaLabel

3.7.2.2.1.6. Proxies

- Utilizar nombres en singular.
- El formato es [Clase]Proxy.
- Usar convenciones de nombres tipo PASCAL.

  Ejemplo:
  UsuarioProxy

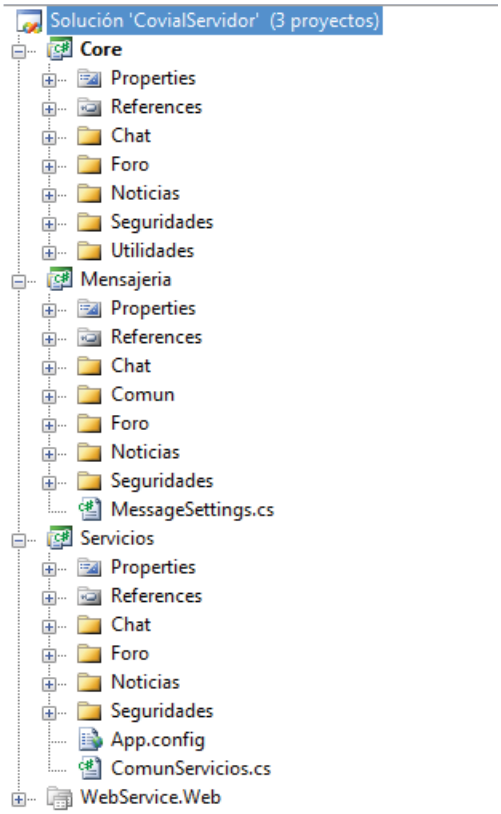
3.7.2.3. SQL SERVER

- Nombres de tablas conjunto con convención de nombres tipo CAMEL.
- Nombres de Columnas conjunto con convención de nombres tipo CAMEL.
- Cada tabla debe tener una llave primaria y si así lo requiere tantas llaves foráneas como las necesite.
- Para integridad de datos, estos nunca son borrados, simplemente son escondidos del Usuario. Este proceso es muy útil para la empresa al poder sacar respaldos reales a cualquier instante de tiempo.

3.7.3. COVIAL WEB 1.0

3.7.3.1. DESARROLLO PROYECTO Covial Servidor

Figura 3.28: Estructura COVIAL SERVIDOR



Fuente: Autores

Se puede observar en la figura que Covial Servidor se encuentra dividido por 3 proyectos y estos en 5 carpetas:

- **Chat:** Dirigidos a la realización de ventanas de Conversación entre usuarios.
- **Foro:** Dirigidos a Compartir archivos, e ideas entre todos los usuarios para comunicación de diferentes temas de interés.
- **Noticias:** Dirigidos a la publicación de Información por parte de COVIAL a los usuarios.
- **Seguridades:** Dirigido a proporcionar seguridad WCF a cada parte del proyecto.
- **Utilidades:** Utilidades comunes usadas por todos los demás módulos.

3.7.3.2. DESARROLLO PROYECTO Covial Cliente.

La realización del Covial Cliente se ve enfocada en las diferentes pantallas que observará el usuario.

Las aplicaciones programadas que tendrán relación con el usuario son las citadas a continuación:

- **Foros:** Representa un segmento de la sociedad donde un grupo de personas mantienen conversaciones acerca de EDUCACIÓN VIAL, otro tema de interés común o bien cualquier tema de actualidad.

- **Catálogos:** Diseñado para dar una visión empresarial de la fundación. Presenta la gran ventaja de poder ser diseñado sin necesidad de programación por parte del usuario. Brindando así flexibilidad al programa al ser modificado en cualquier momento.

- **Chat:** Es una sala de conversación en línea donde los usuarios pueden entablar conversaciones en un mismo bloque.

- **Juegos:** Los juegos podrán ser accedidos siempre y cuando se apruebe un pequeño test con preguntas.

Covial Cliente es realizado mediante una interfaz amigable y muy intuitiva para que el usuario se sienta cómodo.

3.8. IMPLEMENTACIÓN EN LA WEB.

Se genera una publicación tanto del proyecto Covial Cliente y Covial Servidor, donde cada una es generada como publicaciones WEB.

Vale la pena recalcar que los requerimientos para tener en uso a WEB COVIAL1.0 son los citados a continuación:

- Instalación de Framework 3.5. (Para el uso de WCF y SilverLight).
- Internet Information Services de Windows correctamente configurado.
- MIME TYPE en el cual se indique la extensión .**xap** para Silverlight.

Posterior a los requerimientos es necesario subir tal publicación en un dominio virtual en el servidor llamando a los servicios realizados en Covial Servidor.

Los Clientes tendrán acceso a la página COVIAL mediante un browser en internet y tendrá privilegios dependiendo del tipo de Usuario ingresado.

3.9. PRUEBAS.

3.9.1. REALIZACIÓN DE PRUEBAS DIARIAS

Luego de haber concluido la implantación de la aplicación, se espera que los usuarios empiecen con la interacción con el sistema.

Las pruebas se realizan en el periodo de una semana, durante el horario normal de trabajo:

Se pide a los Clientes los cuales visitaron COVIAL en este periodo de tiempo que se utilice el software realizado; A partir de este proceso se espera reparar posibles errores que se puedan producir y a su vez, mejorar cualquier sugerencia brindada.

3.9.2. RESULTADOS DE PRUEBAS

WEB COVIAL 1.0, fue realizado con la visión de entender 100% al usuario, sin embargo, no siempre se puede lograr separar estos dos enfoques (desarrollador y usuario); Por tal razón la primera versión de **WEB COVIAL 1.0** que fue implementada, tuvo pequeños errores que fueron solucionados.

De igual manera se acogió sugerencias para hacer de este proyecto un sistema total, eficiente, y lo más relevante, satisfactorio para el cliente.

A continuación se centran las causas y cambios efectuados a favor de las pruebas realizadas.

- En un inicio, los juegos del Sistema fueron puestos a vista de cualquier visitante, y a sugerencia de uno de estos, se implementó un pequeño test que restringa el ingreso a un determinado juego.

- El programa se realizó en una determinada gama de colores, sin embargo tuvo que ser modificada a solicitud de los directivos de la fundación.

Vale la pena recalcar, como se puede observar, que WEBCOVIAL 1.0 no tuvo errores de relevancia en su ejecución. Esto debido a que, cada módulo del programa, así como los catálogos pueden ser realizados y modificados por parte del Usuario de acuerdo a su propio gusto y conveniencia.

3.10. REUNIONES DE ENTREGA Y USO.

Luego de que el procedimiento haya culminado con éxito, se realiza una reunión con los principales directivos de COVIAL, con el fin de evaluar al Sistema realizado y actualmente en funcionamiento.

De esta reunión **WEB COVIAL 1.0,** obtiene la siguiente evaluación:

3.10.1. EVALUACIÓN FINAL

Concluida la etapa de pruebas, luego de obtener los resultados requeridos y pedidos por COVIAL se puede definir la siguiente evaluación.

Tabla 3.15: Evaluación Final WEB COVIAL 1.0.

Parámetro	Evaluación
Herramienta	El uso de la metodología conocida como UML permite una comprensión del funcionamiento interno del sistema. Por tanto se obtiene un sistema modular y escalable. La utilización de SQL Server, permite tener una base robusta, segura (gracias a cifrado) y rápida.
Administración y mantenimiento	La utilización de estándares de programación permite que el mantenimiento y administración del sistema sea sencillo. Además al usar herramientas conocidas y actuales se encuentra mucha ayuda por parte de la comunidad en Internet.

Parámetro	Evaluación
Seguridad	La aplicación permite un control de acceso en la parte de administración dependiendo del perfil de cada usuario, de acuerdo al cual se aplica las restricciones necesarias para el ingreso de datos, garantizando así, la integridad de los datos. Además las contraseñas se envían de manera cifrada por la red, para prevenir posibles robos de claves, aumentando así la seguridad en la aplicación.
Desempeño	El desempeño es la eficiencia desde el punto de vista de operadores y administradores del sistema. Los directivos de la empresa se encuentran muy satisfechos, ya que el programa ha cumplido sus expectativas: • **En tiempo:** Mejora la toma de decisiones. • **Económico:** Anteriormente la toma de decisiones atrasadas, tenía un coste económico en contra.
Utilización	Los resultados de la fase de pruebas indican que el funcionamiento de la aplicación ha cumplido totalmente con el alcance del proyecto.
Manejo	El sistema fue desarrollado pensando en la comodidad del usuario, de tal manera que sea fácil de manejar.
Contribución	La aplicación COVIAL contribuye a la empresa brindando un servicio: ágil, rápido y eficiente.

Fuente: Autores

4. CONCLUSIONES Y RECOMENDACIONES.

4.1. CONCLUSIONES

- Se puede concluir que la elección de Windows Comunication Foundation para este proyecto fue una buena alternativa de desarrollo ya que brindo un modelo estándar para crear servicios de una forma fácil, rápida y, sobretodo, adaptable. Además una de sus principales ventajas es que no está ligado a IIS (Internet Information Service) de Windows.

- Una de las ventaja más relevante en WEB COVIAL 1.0 es su modularidad por dos razones; La primera es que se redujo el tiempo de trabajo al poder programar dos módulos independientemente, tal como (Base de Datos e Interfaz Gráfica); Segundo, la gran ventaja de poder seguir añadiendo módulos cuantos estos sean necesarios a fin de satisfacer las necesidades del cliente.

- WEB COVIAL 1.0 brinda a la fundación la posibilidad de que varios usuarios naveguen al mismo tiempo en la página, implementando concurrencia en el sistema, por lo que se puede concluir que se ha mejorado la atención y disminuido el tiempo que el usuario emplea en el sistema, permitiendo atender una mayor cantidad de usuarios en un menor tiempo.

- Se puede concluir que el proyecto WEB COVIAL 1.0 cumple a cabalidad los objetivos trazados en el comienzo del proyecto. Satisfaciendo a los clientes y directivos en el uso, funcionamiento y presentación del software. La interfaz gráfica y las nuevas aplicaciones que se ofrecen a los usuarios hacen del programa un producto muy atractivo para el cliente final, de fácil utilización, intuitivo y amigable.

- Como conclusión se puede manifestar que el Identificar los requerimientos del cliente es el paso más importante en el desarrollo del proyecto, puesto que da las pautas para cumplir las metas empresariales al utilizar el software y mejorar la productividad del sistema.

4.2. RECOMENDACIONES

- Se recomienda el desarrollo y la implementación de nuevas características al software, puesto que WEB COVIAL 1.0 es únicamente una muestra de algunas de las capacidades que la tecnología Windows Comunication Foundation puede ofrecer.

- Las herramientas escogidas en este proyecto posiblemente no son las mejores de todas, pero si son las más indicadas para la fundación COVIAL. Se recomienda hacer un análisis mucho más complejo de cada herramienta para garantizar que las escogidas funcionarán tal como lo requiere la empresa.

- Los estándares de XAML, no son difíciles de entender ni de emplear, al igual que los demás lenguajes que Visual .NET proporciona. Se recomienda empezar especificando sus características generales y luego seguir buscando información para ir mejorando su uso dentro de los proyectos de software.

- Es importante mencionar que ninguna red ni programa será 100% seguro. Hasta ahora la fundación COVIAL no ha tenido problemas de manipulación de datos. Sin embargo, es recomendable siempre tener presente este problema, tratando de incorporar cada vez mejores seguridades.

5. BIBLIOGRAFÍA.

- Alfonso Romero B., D. L. (s.f.). Scielo Peru. Recuperado el 10 de Marzo de 2010, de http://www.scielo.org.pe/pdf/iigeo/v10n19/a05v10n19.pdf
- ANDES/GC. (05 de Abril de 2010). *Andes*. Recuperado el 10 de Abril de 2010, de http://andes.info.ec/ecuador/150-accidentes-de-transito-se-produjeron-durante-el-feriado-en-ecuador-10254.html
- Canchala, A. (s.f.). *Microsoft*. Recuperado el 28 de Mazo de 2010, de MSDN: http://msdn.microsoft.com/es-es/library/bb972214.aspx
- DesarrolloWeb. (s.f.). *DesarrolloWeb.* Recuperado el 4 de Junio de 2010, de SQL:

  http://www.desarrolloweb.com/articulos/images/imgpresupuesto/asociacion/tablas.gif
- Fernández, C. A. (s.f.). *Universidad Tecnológica de la Mixteza.* Recuperado el 09 de Febrero de 2010, de Universidad Tecnológica de la Mixteza: http://www.utm.mx/~caff/doc/ModeladoVisualconUML.pdf
- Hoy. (s.f.). *Hoy.* Recuperado el 23 de Enero de 2010, de Regulacion de Transito: http://www.hoy.com.ec/noticias-ecuador/regulacion-del-transito-y-seguridad-287076-
- Kennedy, L. (12 de Febrero de 2007). *I SQL*. Recuperado el 12 de Enero de 2010, de I SQL: http://www.isql.org/2007/02/mysql-vs-sybase-ms-sql-server.html
- Larman, C. (2003). *UML y Patrones – Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* España: Pearson Education.
- M. Luz Cabrera Bernal, D. G. (s.f.). *Universidad de Sevilla.* Recuperado el 4 de Mayo de 2010, de:

  mvc.mfis.googlepages.com/presentacinmvc.ppt

- Microsoft. (s.f.). *Microsoft.* Recuperado el 15 de Abril de 2010, de Visual Studio Requirements:

  http://www.microsoft.com/downloads/details.aspx?FamilyID=75cbcbcd-b0e8-40ea-adae-85714e8984e3&displaylang=en#Requirements

- Microsoft. (24 de Mayo de 2006). *Microsoft SQL Server*. Recuperado el 18 de Diciembre de 2009, de Microsoft SQL Server: http://www.microsoft.com/spain/sql/productinfo/features/compare-features.mspx

- Microsoft. (s.f.). *MSDN.* Recuperado el 2010 de Abril de 01, de Centro de Desarrollo Silverlight: http://msdn.microsoft.com/es-es/silverlight/default.aspx

- Microsoft. (s.f.). *MSDN*. Recuperado el 01 de Mayo de 2010, de Conceptos básicos de Windows Communication Foundation: http://msdn.microsoft.com/es-es/library/ms731079(v=VS.90).aspx

- Microsoft. (s.f.). *MSDN*. Recuperado el 05 de Mayo de 2010, de Requisitos del sistema WCF: http://msdn.microsoft.com/es-es/library/ms733890.aspx

- Microsoft. (s.f.). *MSDN- Windows Communication Foundation.* Recuperado el 30 de Marzo de 2010, de MSDN: Windows Communication Foundation

- Microsoft. (2010). *Visual Studio*. Recuperado el 25 de Febrero de 2010, de MSDN: http://msdn.microsoft.com/es-es/vstudio/2010.aspx

- Morelos, U. A. (s.f.). *Universidad Autónoma del Estado de Morelos.* Recuperado el 05 de Junio de 2010, de SQL SERVER: www.uaem.mx/posgrado/mcruz/cursos/miic/sqlserver2.ppt

- Nominado, C. (s.f.). *Conductor Nominado.* Recuperado el 25 de Noviembre de 2009, de Conductor Nominado: http://www.conductornominado.com.ec/para-amigos/estadisticas.html

- Oracle. (s.f.). *Oracle Database 10g Express Edition*. Recuperado el 2009 de Diciembre de 17, de Oracle Database 10g Express Edition: http://www.oracle.com/lang/es/database/Express_Edition.html
- Ovar Jacobson, G. B. (2000). *El Proceso Unificado de Desarrollo de Software.* España: Addison Wesley.
- Pressman, R. (2002). *Ingeniería de Software.* Madrid: MacGraw Hill.
- proactiva-calidad. (s.f.). *proactiva-calidad.* Recuperado el 5 de Febrero de 2010, de Patrón "Modelo-Vista-Controlador": http://www.proactiva-calidad.com/java/patrones/mvc.html
- Scribd. (s.f.). *Scribd.* Recuperado el 18 de Febrero de 2010, de Modelo Vista Controlador: http://www.scribd.com/doc/17677638/mvc
- Software, T. (01 de Agosto de 2004). *MySQL vs. SQL Server*. Recuperado el 20 de Diciembre de 2009, de TOMETA Software: http://www.tometasoftware.com/mysql_vs_sqlserver.asp
- Torres, P. L. (s.f.). *Universidad Politécnica de Valencia.* Recuperado el 05 de Febrero de 2010, de Universidad Politécnica de Valencia: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/UML_Historia_Intro.pdf
- Wikipedia. (12 de Mayo de 2010). *Wikipedia*. Recuperado el 18 de Mayo de 2010, de http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
- Wyrm. (14 de Septiembre de 2009). *CRISTALAB*. Recuperado el 01 de Abril de 2010, de http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-en-visual-basic-.net-c273l/

6. ANEXO

6.1 ANEXO 1

XAML

FUENTE: http://msdn.microsoft.com/es-es/library/cc189054(VS.95).aspx

Definición XAML

XAML (Lenguaje de Marcado de Aplicaciones Extensible), es un lenguaje de para la programación de aplicaciones declarativa. Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF) y Silverlight, utilizan este tipo de lenguaje proporcionando compatibilidad en el lenguaje de marcado de aplicaciones extensible (XAML) para los tipos de Windows Presentation Foundation (WPF), Windows Communication Foundation y SilverLight, de tal manera que se puede crear una Interfaz gráfica de Usuario (GUI) mucho más rica en tecnología.

Una de las grandes ventajas que presenta esta tecnología, es la versatilidad de trabajar con entornos WEB tan bien como entornos de Escritorio.

Este lenguaje XAML es un lenguaje declarativo el cual usa un archivo de código subyacente independiente para responder a los eventos y manipular los objetos declarados en XAML.

XAML al ser un lenguaje declarativo basado en XML es muy intuitivo para crear interfaces desde el prototipo hasta la producción, sobre todo para personas con conocimientos de diseño y tecnologías web.

Los nombres de los elementos y atributos XAML distinguen mayúsculas de minúsculas. Por ejemplo, si el valor de un atributo declara un nombre de miembro de enumeración, el comportamiento integrado que convierte una cadena de nombre de miembro para devolver el valor del miembro de enumeración no distingue mayúsculas de minúsculas. En cambio, el valor de la propiedad Name, así como los métodos de utilidad para trabajar con objetos

basados en el nombre declarado por la propiedad Name, distinguen mayúsculas de minúsculas a la hora de considerar la cadena de nombre.

Silverlight y XAML se integran a la perfección juntos. Cada elemento XAML puede ser accedido o manipulado del mismo lado de cliente que actuaría con cualquier elemento de HTML. Entonces se puede usar JavaScript en un código separado - detrás del archivo para responder a acontecimientos y manipular los objetos que usted declaró en XAML en caso de Silverlight 1.0 y C */VB en caso de Silverlight 1.1.

XAML (.xaml el archivo) puede ser aumentado con el código - **detrás del código**, escrito en cualquier lengua .NET, que contiene la lógica de programa. Esto puede ser usado a tanto para uso Silverlight como para páginas de HTML que recibe el control de Silverlight.

Silverlight posee una biblioteca de clase de peso ligero que destaca, entre otros, mandos extensibles, XML Servicios de Web, componentes conectados a una red y LINQ APIs. Esta

biblioteca de clase es un subconjunto de y es bastante más pequeña que la Biblioteca de Clase Baja del Marco .NET. Algunas clases apoyadas que ser manejo de cuerda, expresiones regulares, entrada y salida, reflexión, colecciones, y globalización.

Algunas propiedades solo pueden establecerse mediante la sintáxis de elementos de propiedad. Para usar la sintáxis de elementos de propiedad, debe ser posible especificar una nueva instancia de un elemento de objeto a fin de "rellenar" el valor de elemento de propiedad.

Para usar la sintáxis de elementos de propiedad, debe crear elementos XML para la propiedad que desee establecer. Estos elementos tienen el formato <object.property>. En XML estándar, este elemento se consideraría simplemente un elemento con un punto en su nombre. Sin embargo, en XAML,

el punto en el nombre de elemento identifica el elemento como un elemento de propiedad, con property como propiedad de object.

<objeto>

<objeto.propiedad>

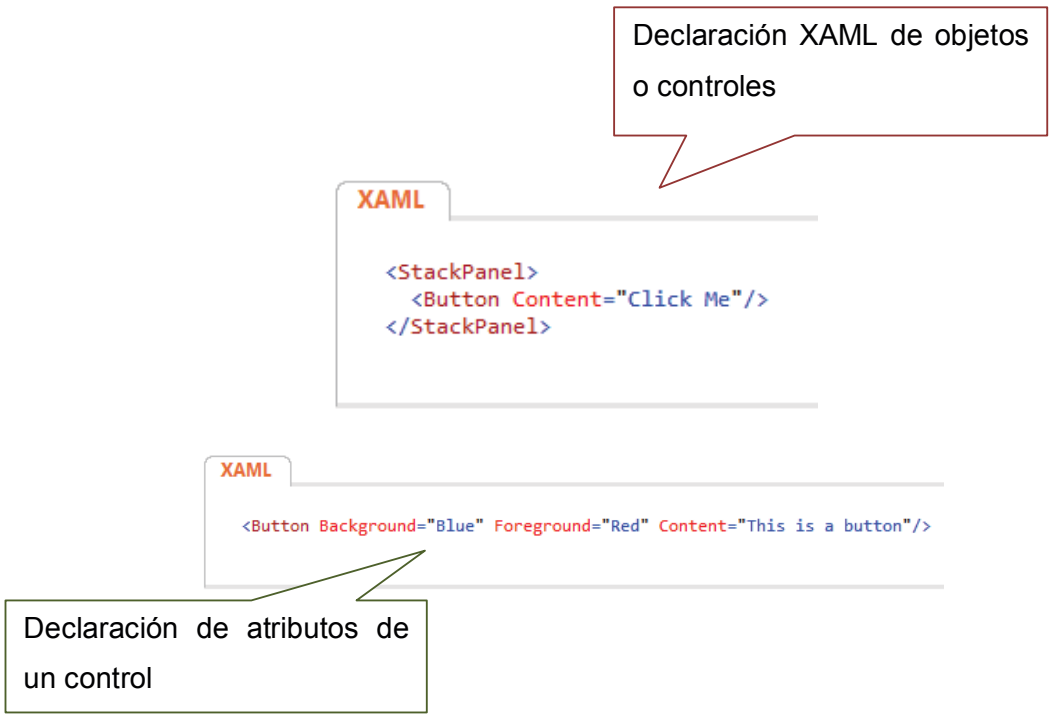
valorDePropiedadComoElementoDeObjeto

</objeto.propiedad>

</objeto>

EJEMPLOS XAML:

Figura 6.1 Ejemplos XAML



XAML

```
<Button>
  <Button.Background>
    <SolidColorBrush Color="Blue"/>
  </Button.Background>
  <Button.Foreground>
    <SolidColorBrush Color="Red"/>
  </Button.Foreground>
  <Button.Content>
    This is a button
  </Button.Content>
</Button>
```

Declaración de Propiedades

Fuente: http://msdn.microsoft.com/es-es/library/cc189054(VS.95).aspx

6.2 Anexo 2

Manual de Instalación

Requisitos

Características de Hardware

Mínimos Requerimientos:

- Procesador de 32 bits (x86) o 64 bits (x64) a 1 gigahercio (GHz) o más.
- Memoria RAM de 1 gigabyte (GB) (32 bits) o memoria RAM de 2 GB (64 bits).
- Espacio disponible en disco rígido de 16 GB (32 bits) o 20 GB (64 bits)

Características de Software

Sistemas Operativos Soportado:

Windows Server 2003 Standard Edition, Windows Server 2003 Enterprise Edition, Windows Server 2003 Small Business Edition, Windows Server 2008 Standard, Windows Server 2008 Enterprise Edition.

Base de Datos:

Microsoft SQL Server 2008 Standart Edition, Microsoft SQL 2008 Enterprise Edition.

Programas:

Microsoft Framework 3.5

Microsoft Framework 3.5 Service Pack 1

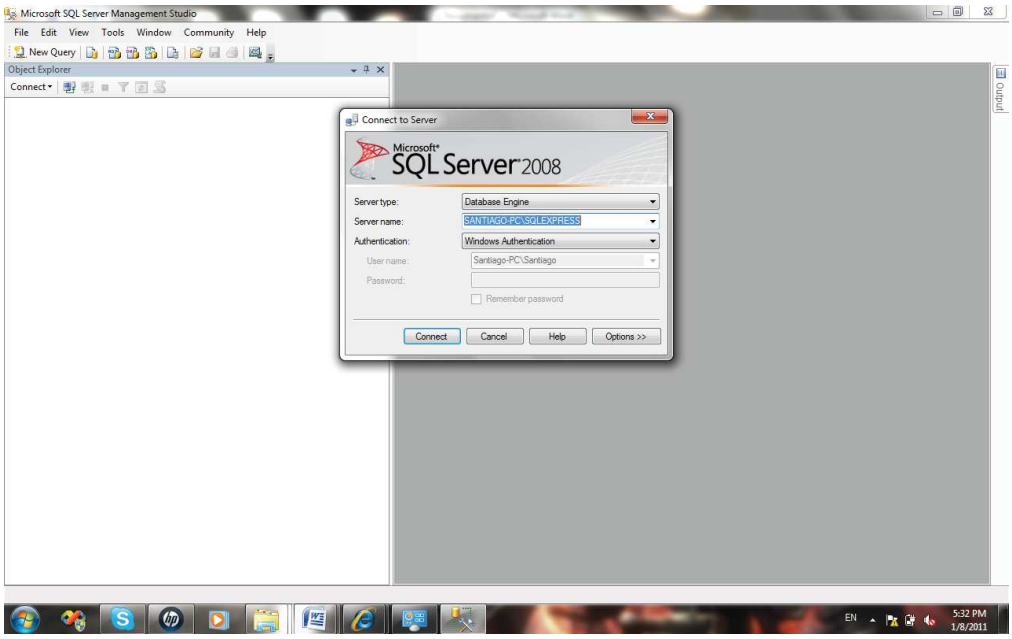
Internet Information Services (IIS) 6.0 o superior

Instalación.

1. Verificar que se encuentre instalado en el servidor los siguientes programas:
    - Internet Information Services (IIS)
    - Framework v 3.5 con el Service Pack 1
    - Microsoft SQL Server 2008

2. Abrimos el programa “SQL Server Management Studio” y nos conectamos al motor de base de datos creado el momento de la instalación.

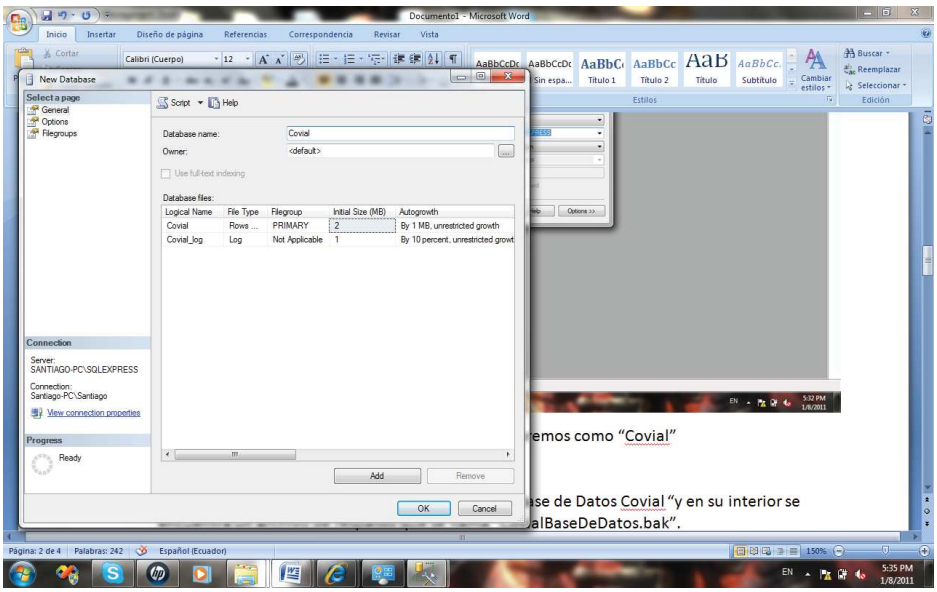
Figura 6.2: Abriendo SQL Server Management Studio



Fuente: Autores

3. Creamos una nueva base de datos la cual la nombraremos como “Covial”

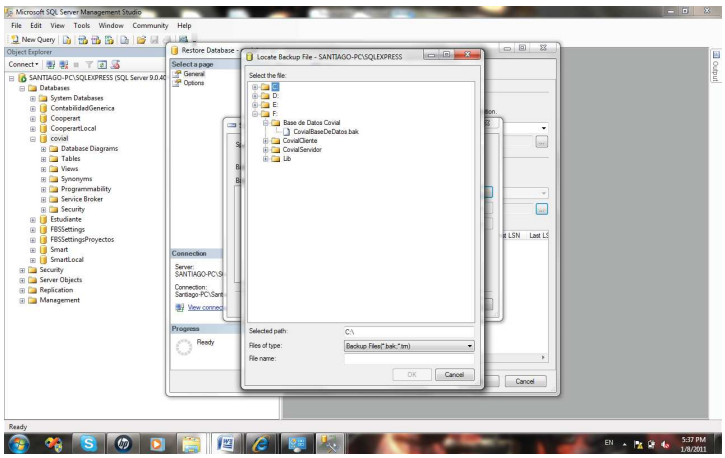
Figura 6.3: Creando Base de Datos



Fuente: Autores

4. En el CD de instalación existe una carpeta llamada “Base de Datos Covial “y en su interior se encuentra un archivo de respaldo que se llama “CovialBaseDeDatos.bak”, este archivo es el respaldo de la base de datos original (Sin Datos), procedemos a restaurar el respaldo en nuestra base creada (Covial).

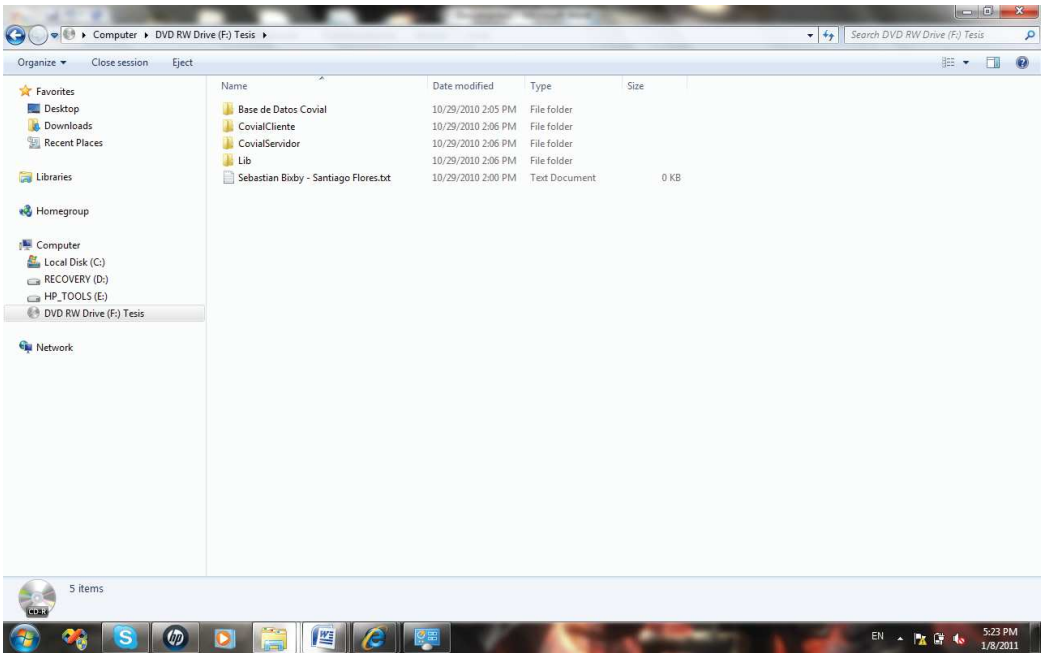
Figura 6.4: Restaurando Base de Datos Covial



Fuente: Autores

5. En el CD de instalación se encuentra 2 carpetas, CovialCliente (publicación de Cliente) y CovialServidor (publicación del Servidor).

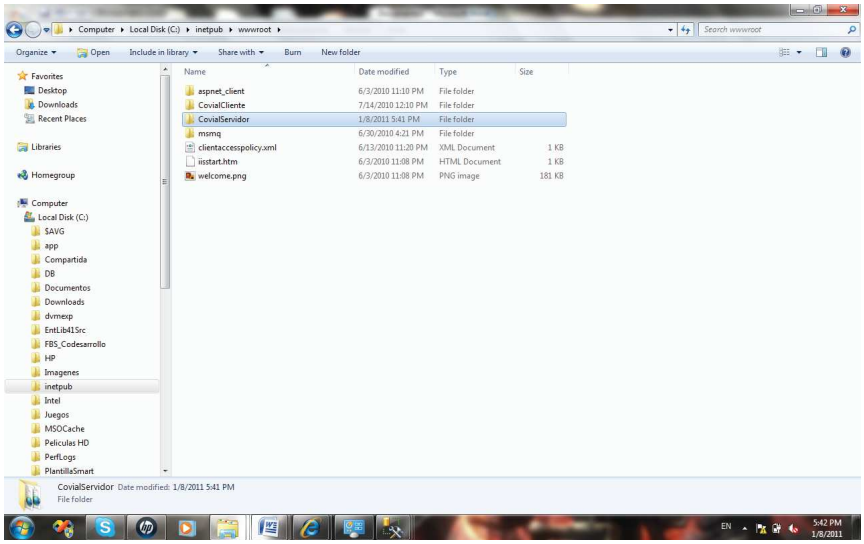
Figura 6.5: Consultando CD de Instalación



Fuente: Autores

6. Copiamos estas dos Carpetas a la carpeta inepta/wwwroot ubicada en el disco C (Esta carpeta se crea una vez instalado el Internet Information Services).

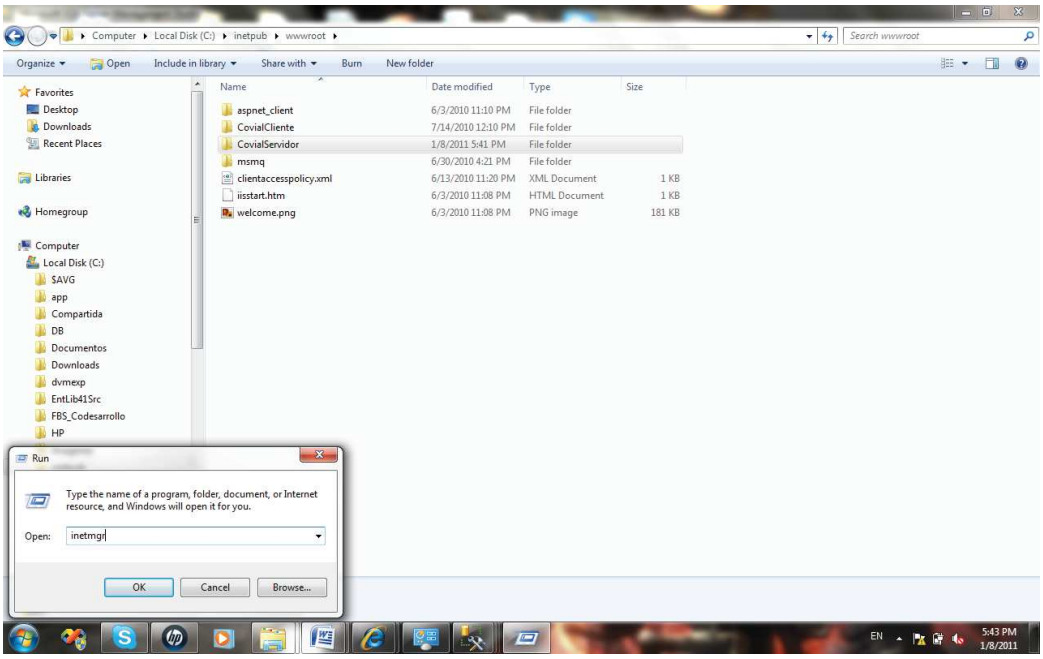
Figura 6.6: Copiando archivos de instalación



Fuente: Autores

7. Ingresamos al Internet Information Services utilizando el comando "inetmgr" en Inicio>Ejecutar

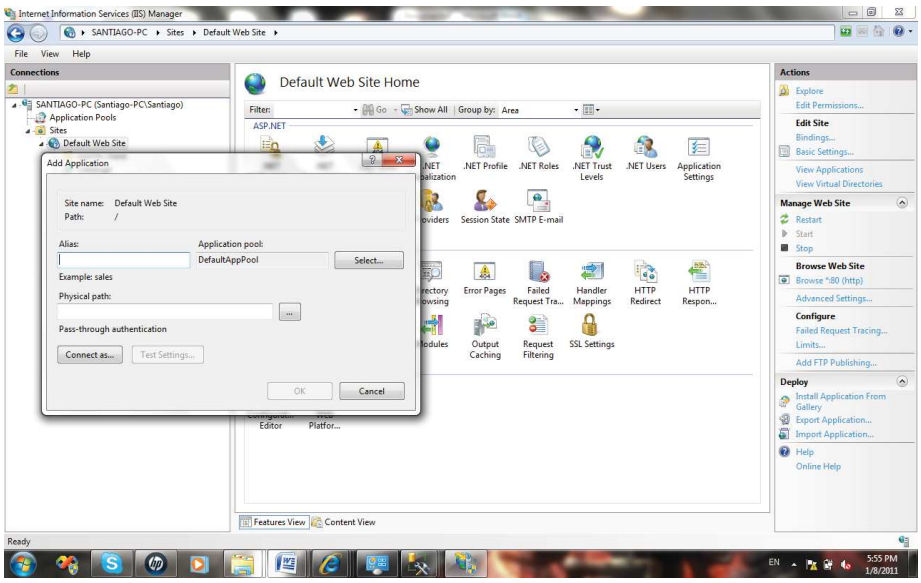
Figura 6.7: Abriendo Internet Information Services



Fuente: Autores

8. Creamos un nueva Aplicación para cliente y para servidor

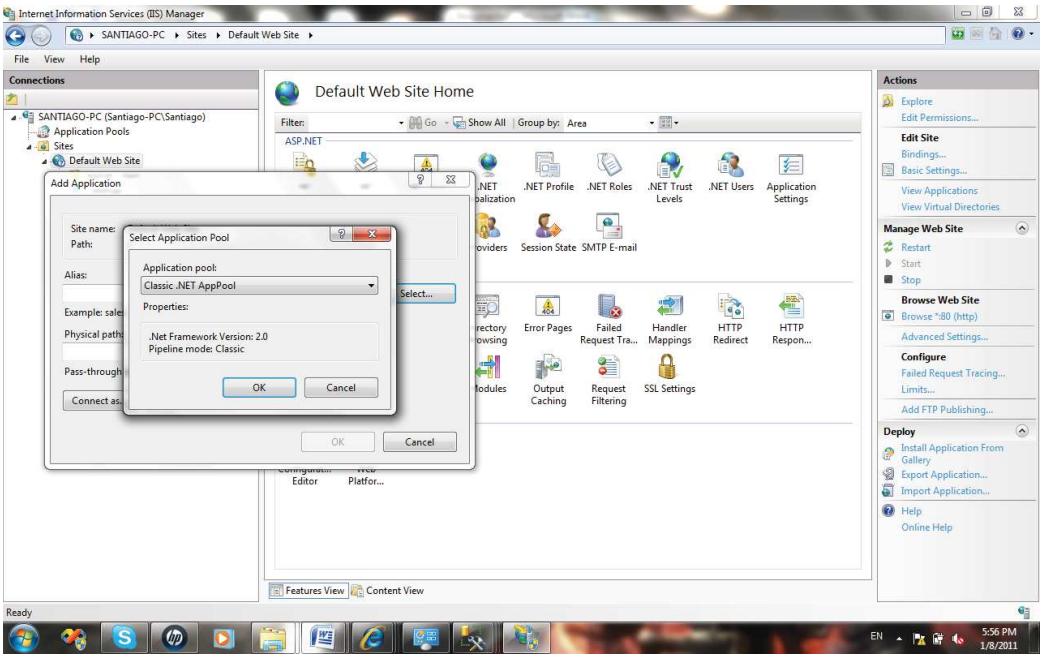
Figura 6.8: Crear aplicación web en el IIS



Fuente: Autores

9. En application pool escogemos la opción “Classic .NET AppPool” este contiene todas las funcionalidades del Framework 3.5 que es la recopilación del Framework 2.0 y del Framework 3.0 mas alguna nuevas características.

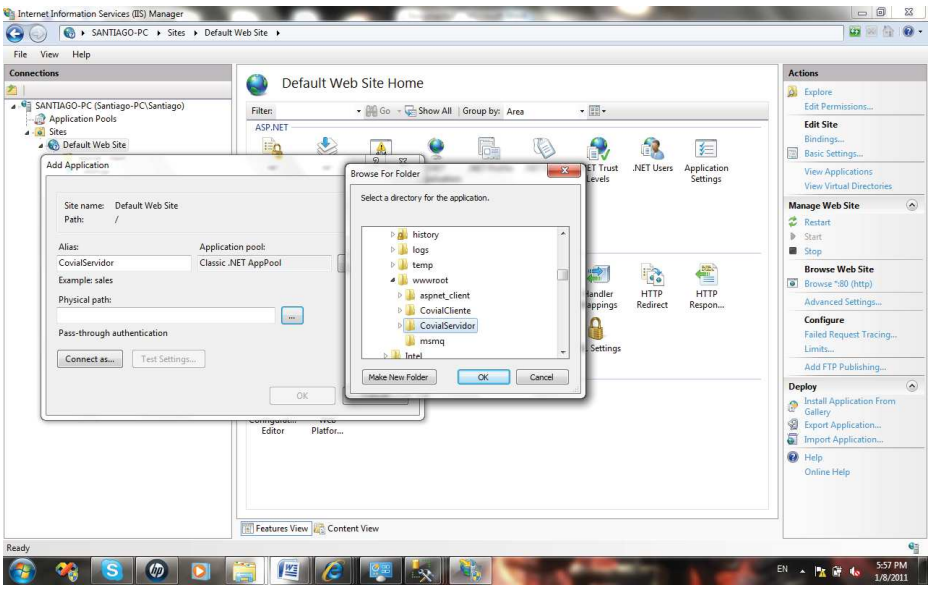
Figura 6.9: Escogiendo Application Pool



Fuente: Autores

10. Ponemos en alias CovialServidor y Covial Cliente respectivamente e indicamos a que carpeta de pertenecen.

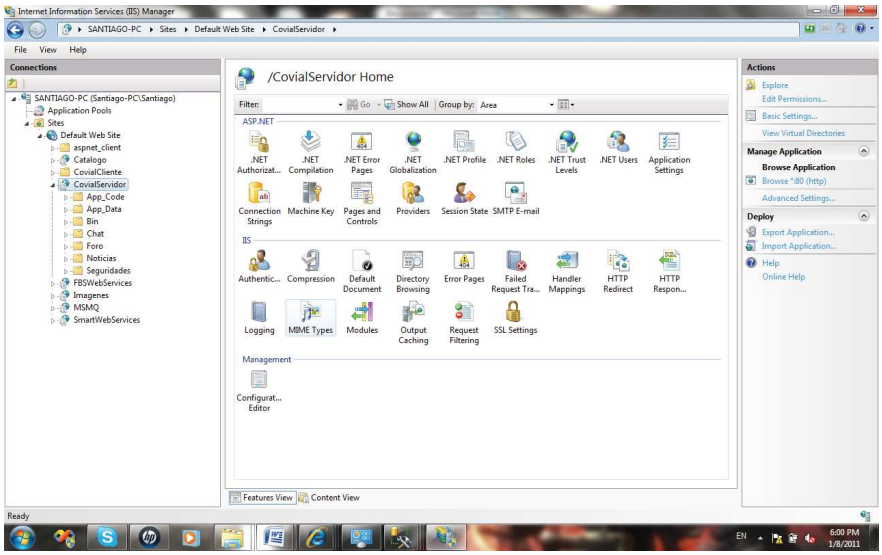
Figura 6.10: Nombrando a la aplicación web



Fuente: Autores

11. Una vez creado nuestras 2 aplicaciones en el Internet Information Services, seleccionamos una por una y nos dirigimos a “Mime Types”

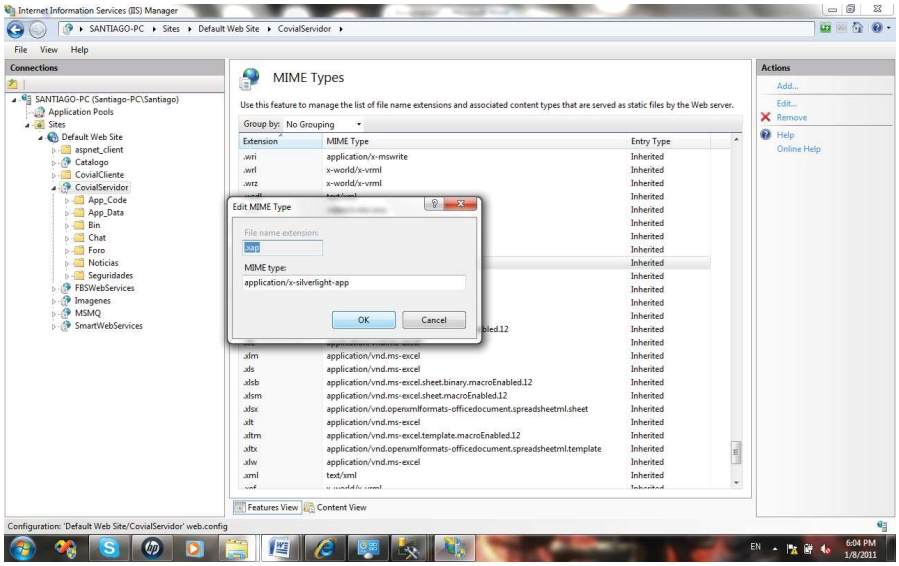
Figura 6.11: Escogiendo la opción “Mime Types”



Fuente: Autores

12. Una vez ingresados en la opción Mime Types, creamos un nuevo tipo, en el campo extensión digitamos “.xap” y en campo MIME Type digitamos “aplicación/x-silverlight-app”

Figura 6.12: Configurando Mime Types

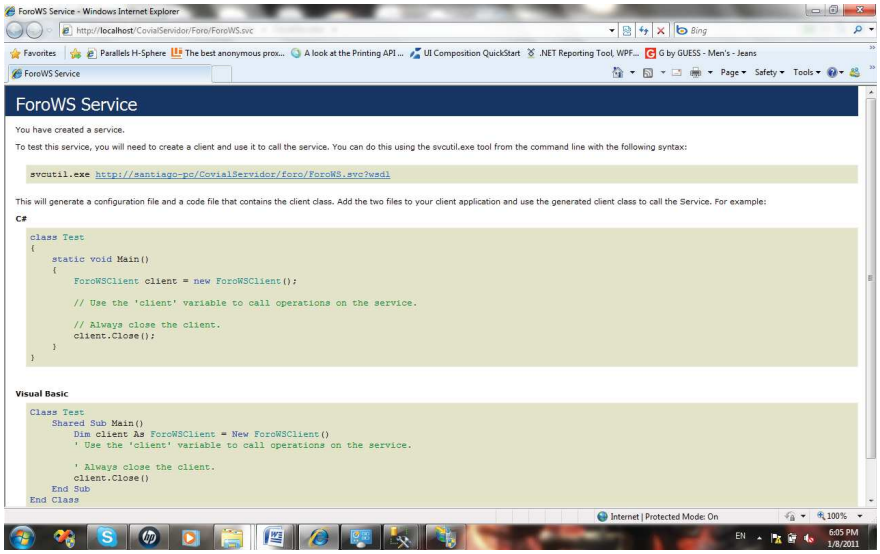


Fuente: Autores

13. Listo en este momento ya puedes utilizar el Portal Web Covial 1.0

Servidor:

Figura 6.13: Probando la aplicación



Cliente:

Fuente: Autores

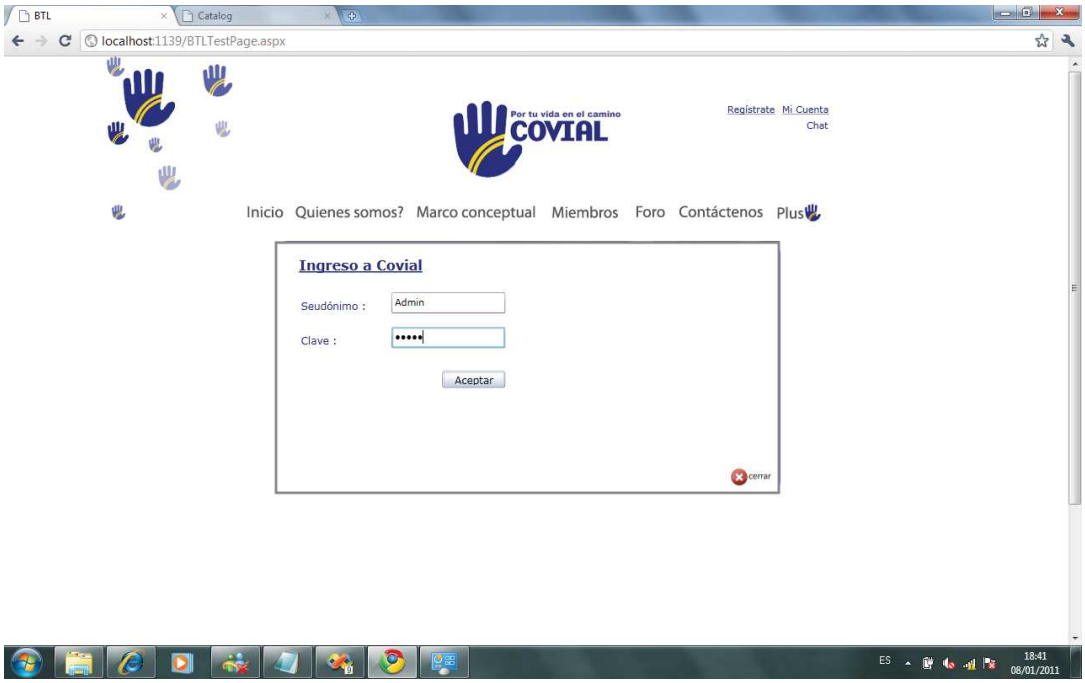
6.3 Anexo 3

Manual de Usuario Administrador

Sitio Web Covial

Lo primero que se debe realizar es el ingreso como usuario Administrador para eso debemos ingresar al menú y hacer click en “Mi Cuenta”, este nos desplegara una nueva pantalla en la cual podemos ingresar nuestro usuario y contraseña.

Figura 6.14: Ingreso al Perfil del usuario

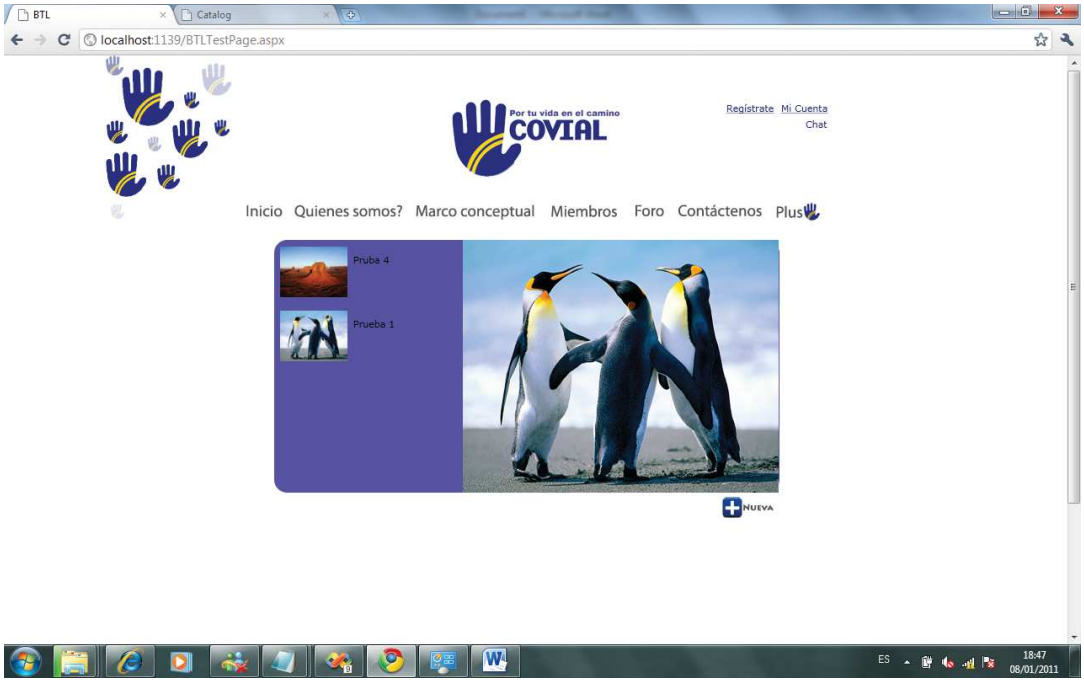


Fuente: Autores

Noticias

Una vez ingresado al sistema como usuario administrador nos colocamos en el menú inicio el cual nos desplegara las noticias que podremos dar a conocer, en la parte inferior derecha de las noticias desplegadas estará habilitado un botón que nos permitirá crear una nueva noticia.

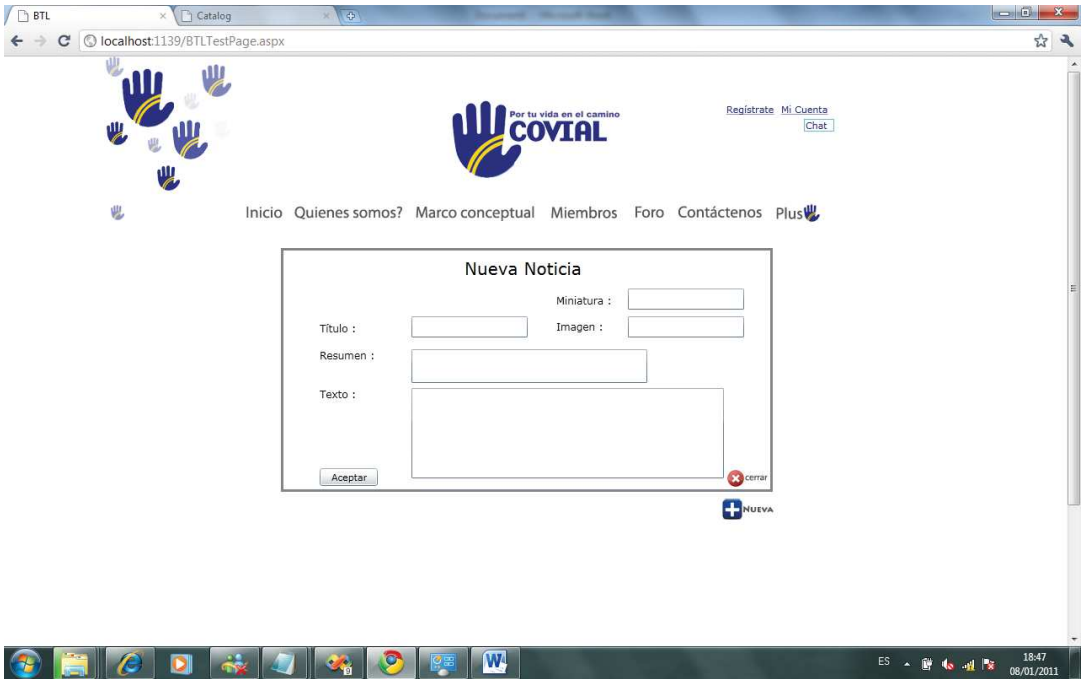
Figura 6.15: Creando nueva noticia



Fuente: Autores

La pantalla de ingreso de la noticia estará formada por un titulo, un resumen, una descripción, una imagen, y una imagen en miniatura o (Thumbnail).

Figura 6.16: Creando Noticia



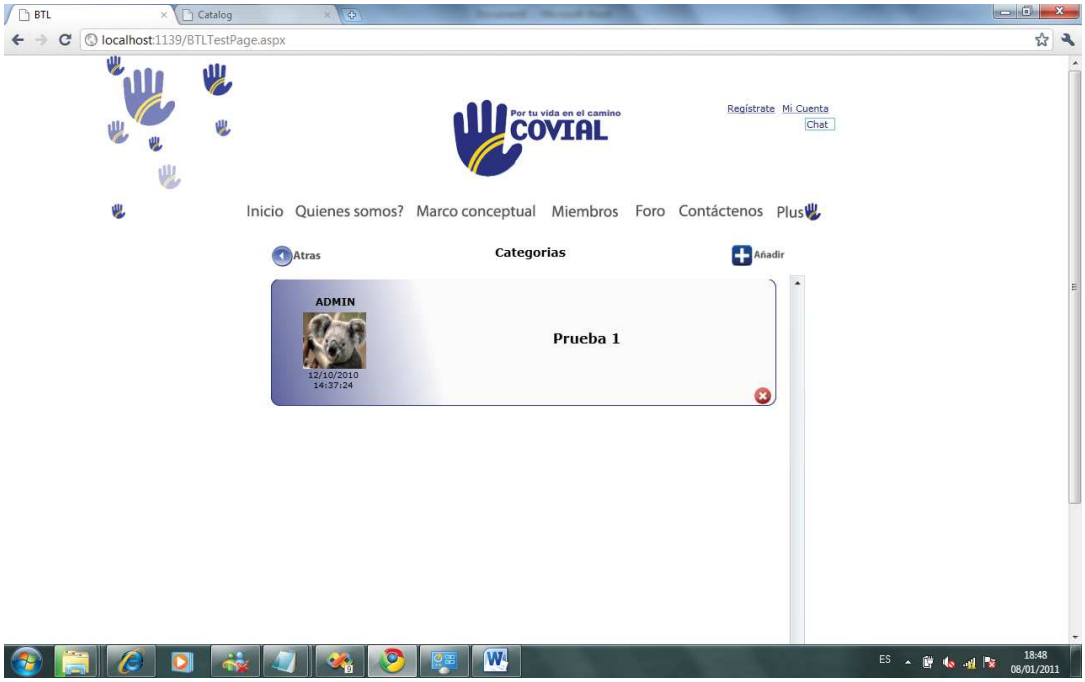
Fuente: Autores

Foro

En el menú “Foro” podemos visualizar todos los comentarios y dudas de nuestros clientes van ingresando día a día.

Una de nuestras principales funciones es crear categorías y temas de debate para que nuestros clientes pueden ingresar a los mismo y manifestar su opinión.

Figura 6.17: Ingreso al foro



Fuente: Autores

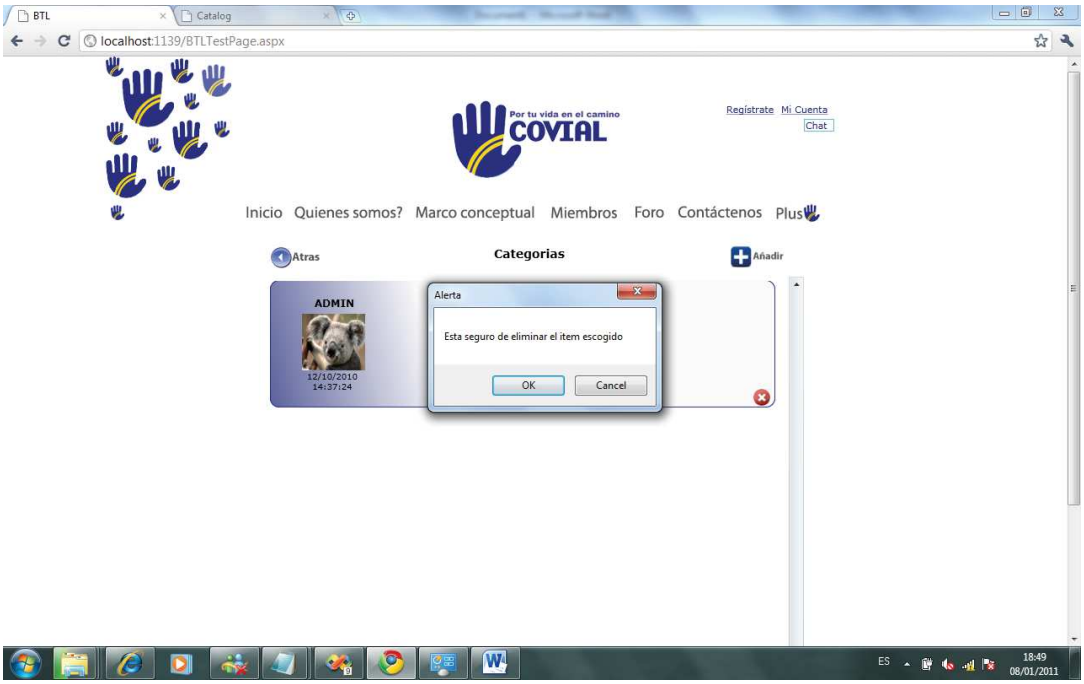
En la esquina inferior derecha del foro tenemos habilitado un botón de "Creación", este nos permitirá crear una nueva Categoría, Tema o respuesta.

Figura 6.18: Creando categoría

Fuente: Autores

El sistema Covial 1.0 nos permite moderar todas las Categorías, Temas y respuestas de nuestro foro logrando así eliminar todo contenido que infrinja las políticas de utilización del foro establecidas.

Figura 6.19: Eliminando ítem del foro



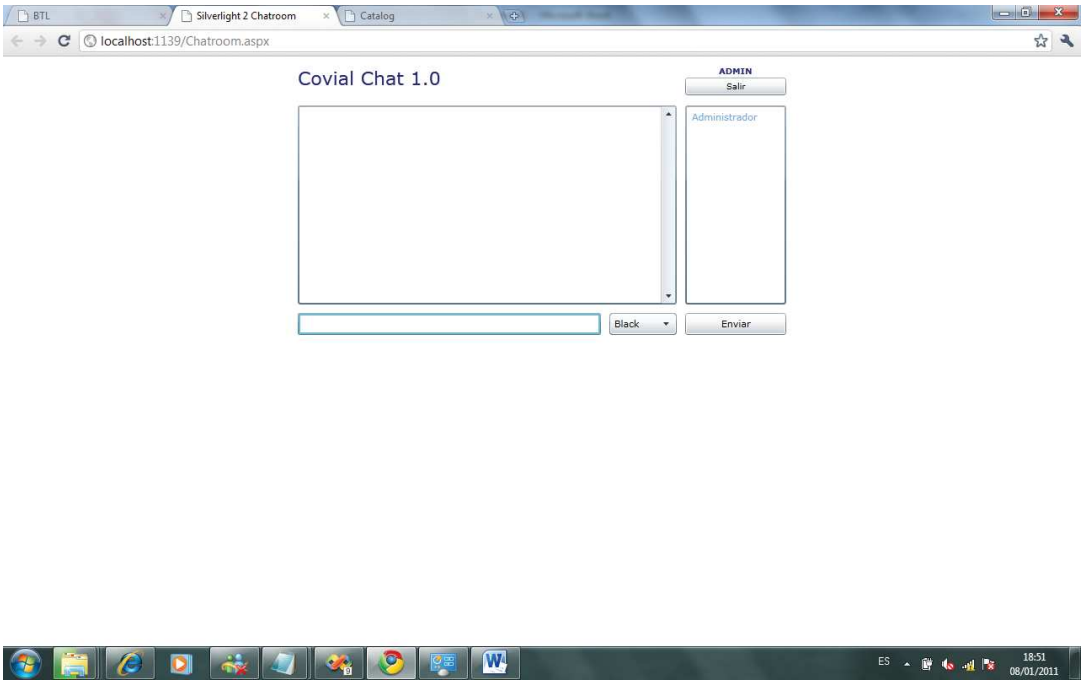
Fuente: Autores

Chat

Al brindar un servicio directo e inmediato a nuestros clientes el manejo exclusivo de un chat en tiempo real es necesario.

Para ingresar al chat y establecer sus dudas el cliente debe ingresar con su usuario y clave desde la página de inicio del foro.

Figura 6.20: Ingreso al chat



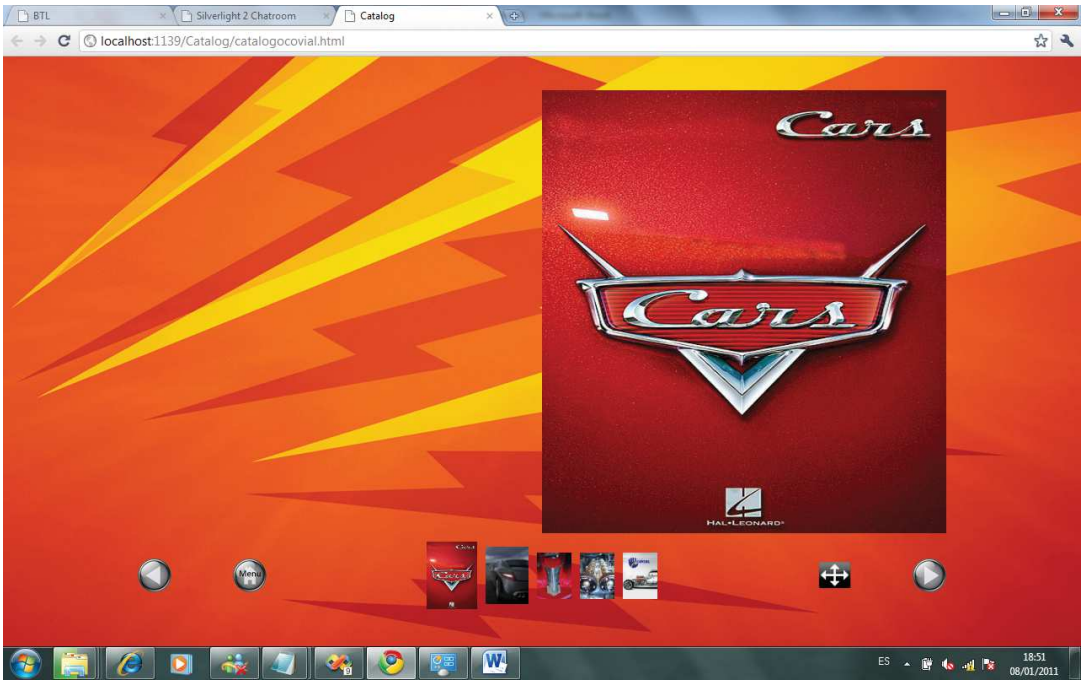
Fuente: Autores

Catálogo covial 1.0

Ha diferencia de nuestro sitio web el catalogo no funciona con una base de datos si no está creado para una rápida creación y modificación del mismo.

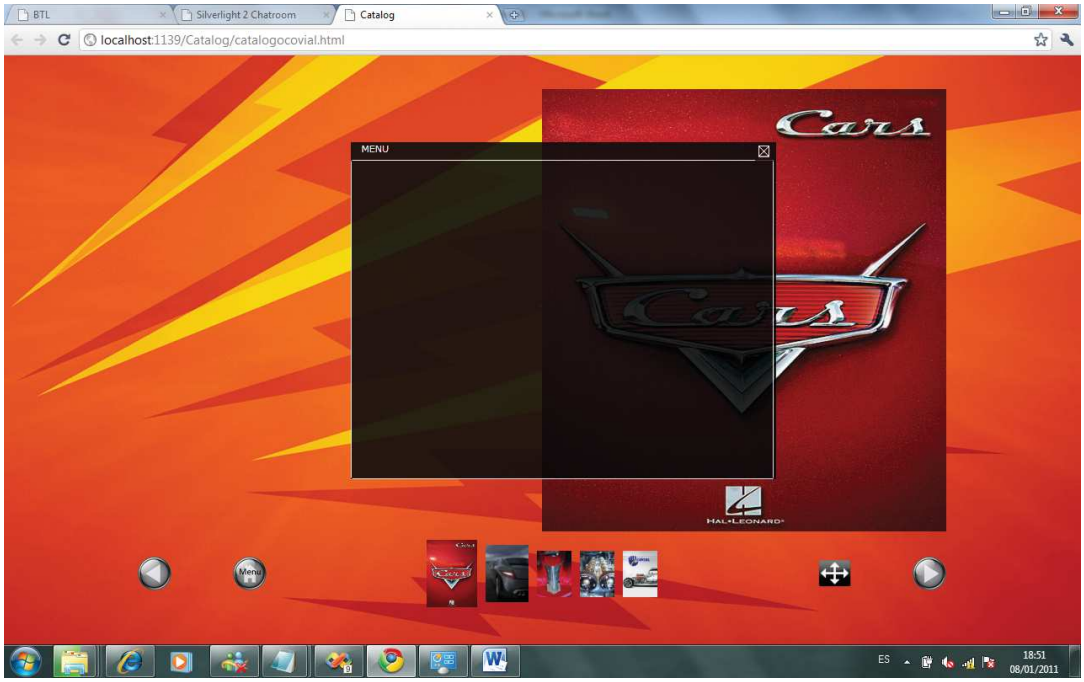
El catalogo posee de un libro en 3D el cual es el núcleo del catalogo, una barra de navegación inferior tipo carrusel, y un menú.

Figura 6.21: Ingreso al catálogo



Fuente: Autores

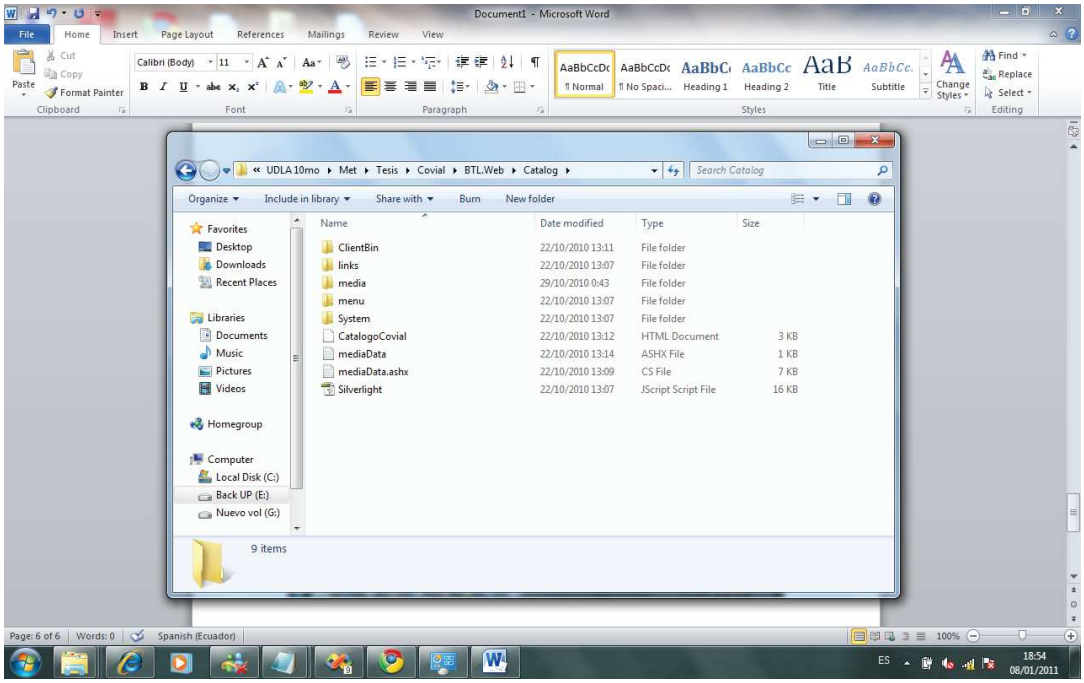
Figura 6.22: Ingreso al menú



Fuente: Autores

En la carpeta de cliente en donde se encuentra la publicación de cliente, accedemos a la carpeta Catalog

Figura 6.23:Editando el catálogo



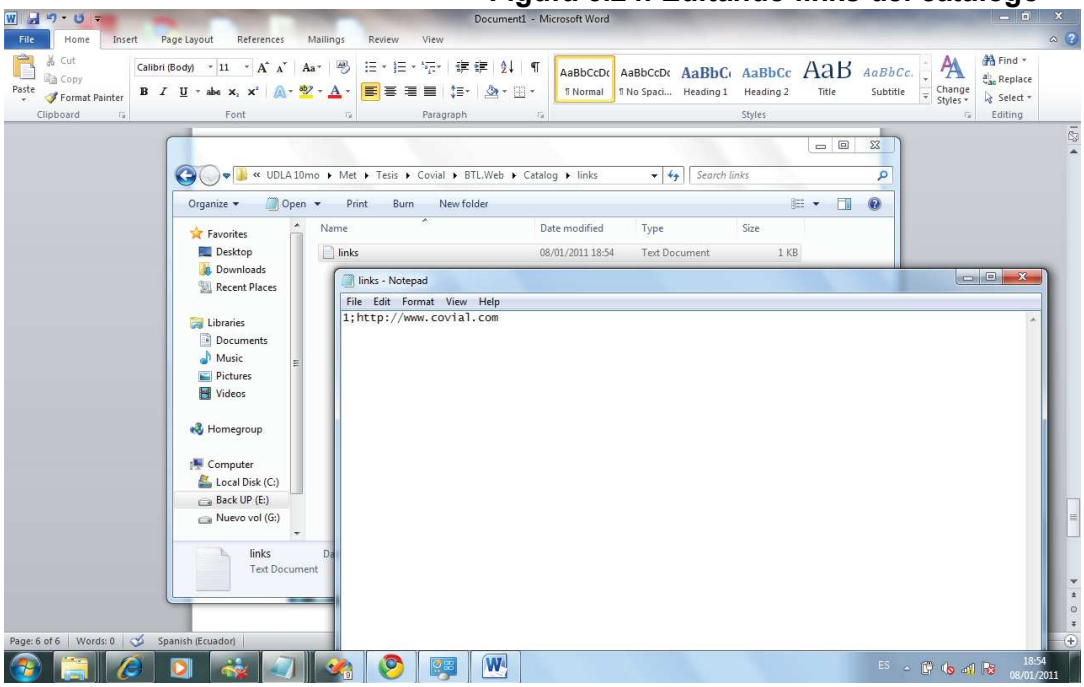
Fuente: Autores

Esta carpeta contiene las siguientes sub carpetas de administración del catalogo

- Links
- Media
- Menú
- System

Links.- En esta subcarpeta se encuentra un archivo de texto (Links.txt) en el cual podemos ingresar links a cualquiera de la hojas del catalogo.

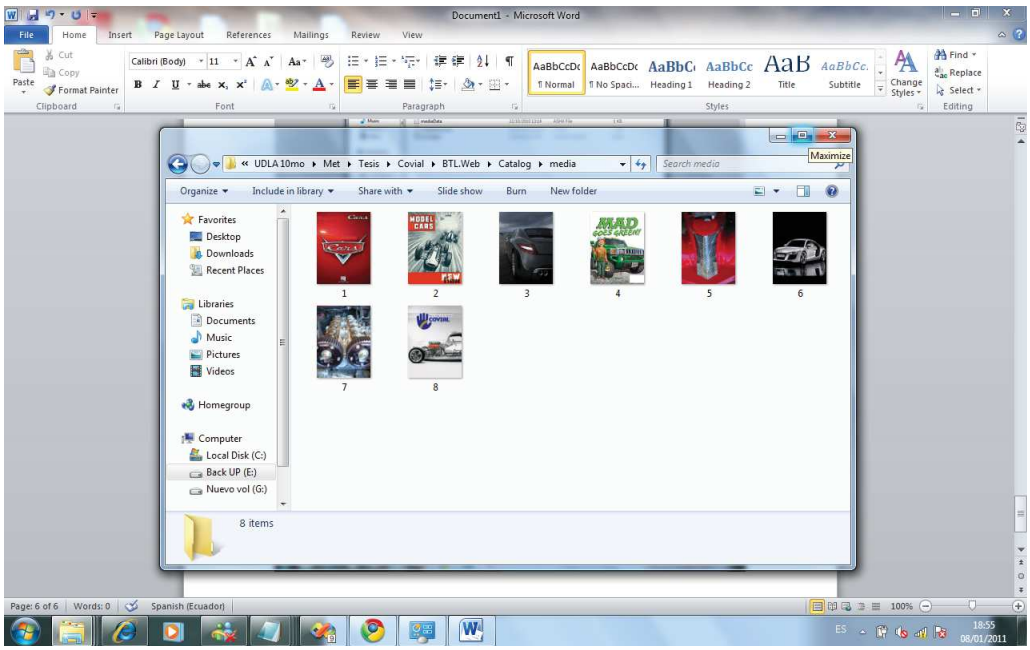
Figura 6.24: Editando links del catálogo



Fuente: Autores

Media.- Esta subcarpeta posee todo el contenido de nuestro catálogo 3D y de nuestro carrusel (Imágenes y Videos) las cuales se ordenaran de acuerdo al número ingresado en el nombre del archivo de manera ascendente. Las extensiones soportadas para imágenes son jpg y png, para los videos la extensión soportada es wmv.

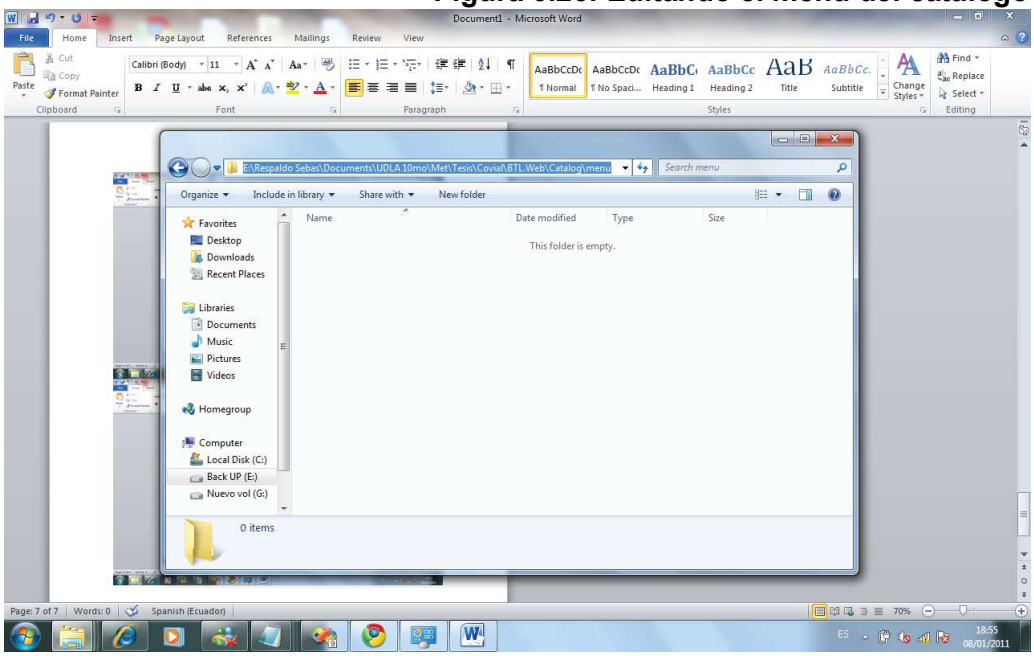
Figura 6.25:Editando las hojas del catálogo



Fuente: Autores

Menú.- Esta subcarpeta posee las imágenes que aparecerán en el menú las cuales son accesos rápidos de navegación en nuestro catalogo Ej. Queremos que la hoja numero 14 posea un acceso rápido dentro del menú entonces copiamos la imagen 14 de la carpeta media a la carpeta menú y listo.

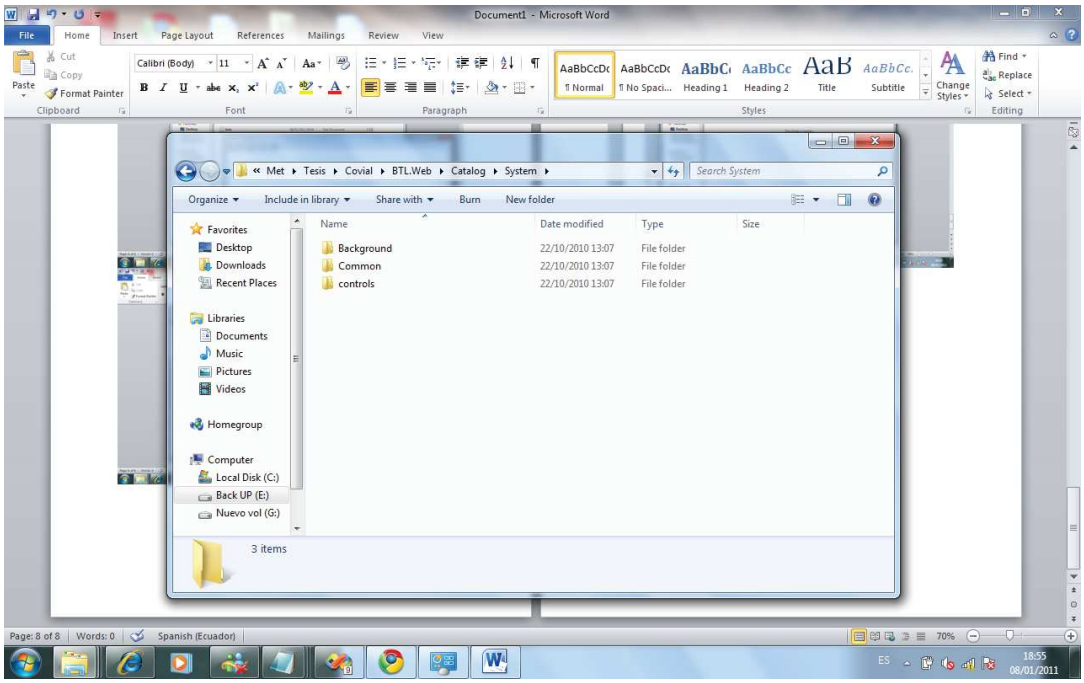
Figura 6.26: Editando el menú del catálogo



Fuente: Autores

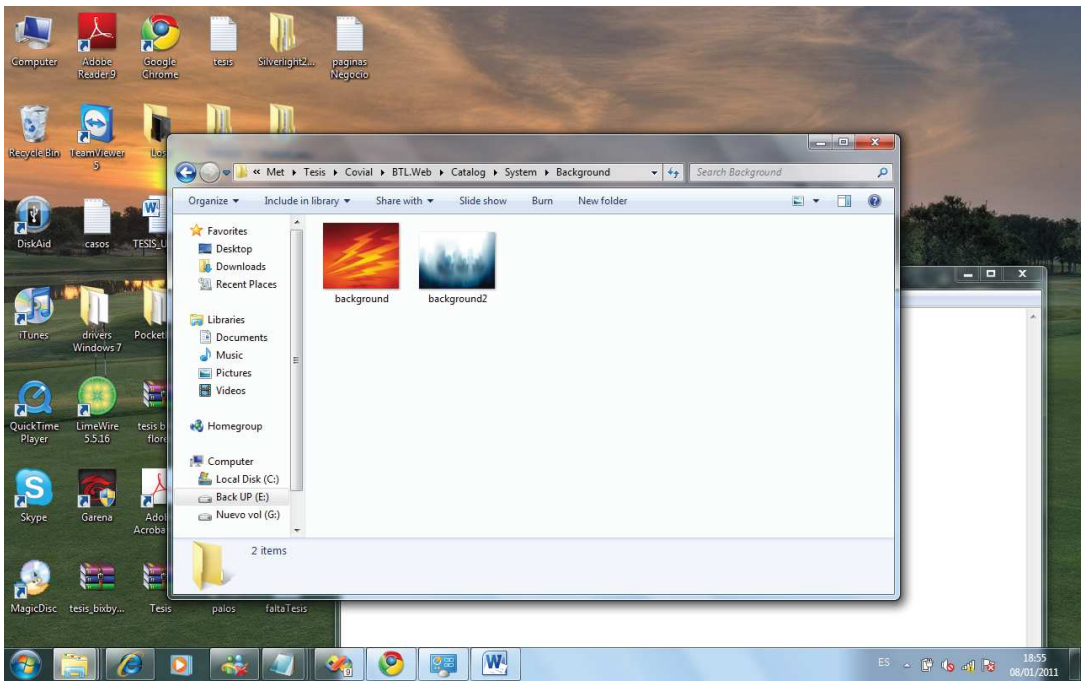
System.- En esta subcarpeta se encuentran

Figura 6.27: Editando el catálogo



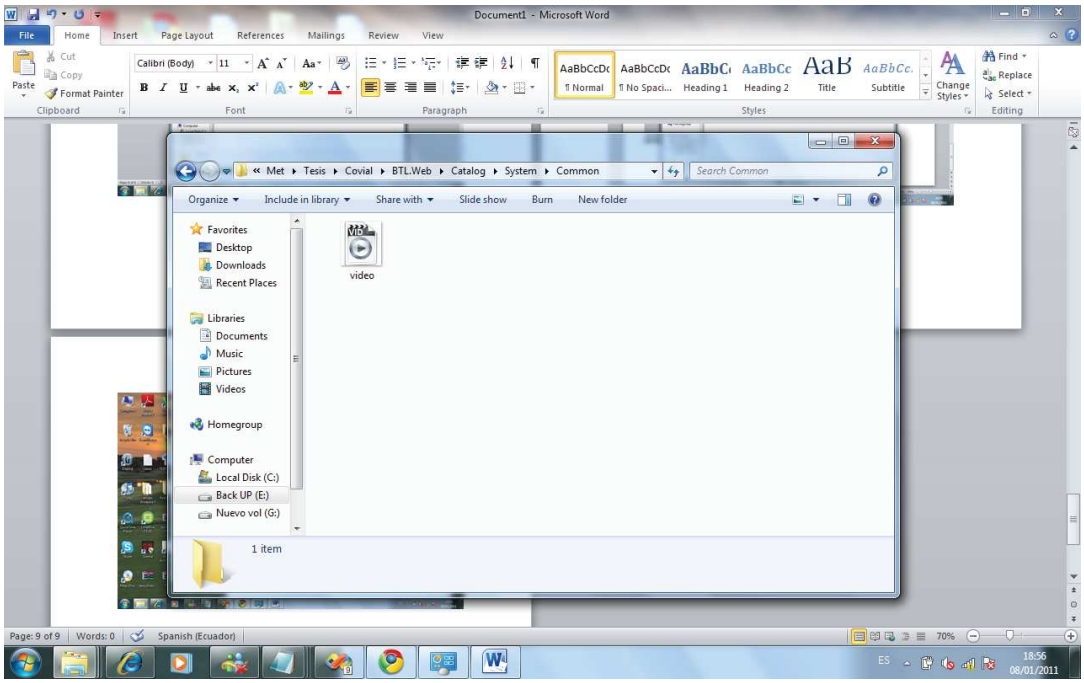
Fuente: Autores

Figura 6.28: Editando el fondo del catálogo



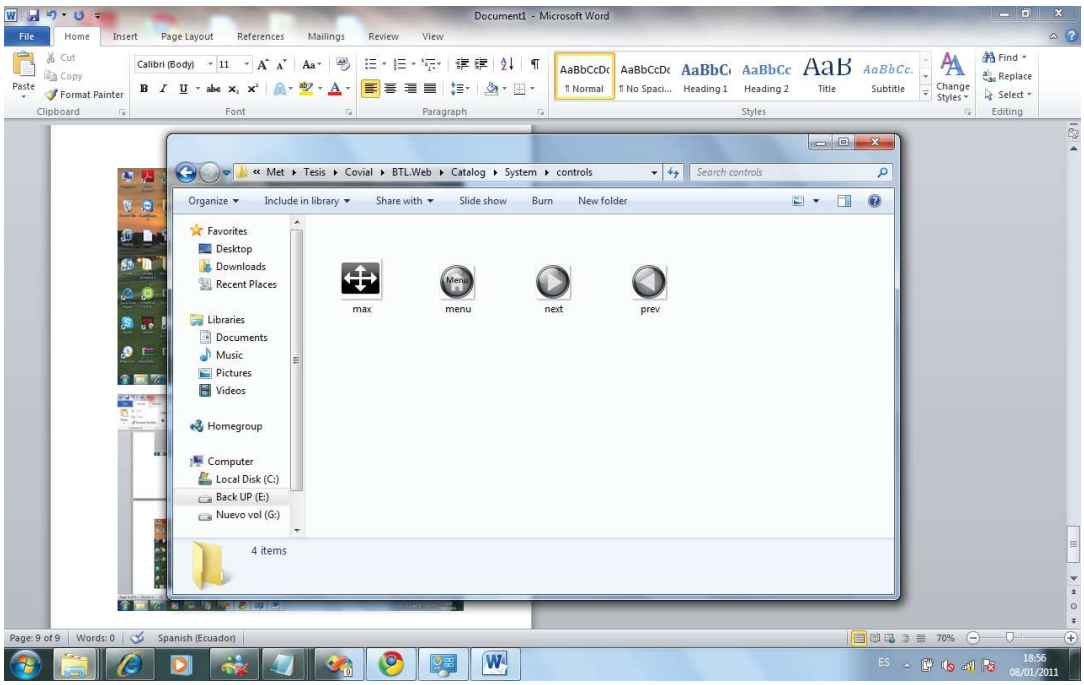
Fuente: Autores

Figura 6.29: Editando videos del catálogo



Fuente: Autores

Figura 6.30: Editando los botones del catálogo



Fuente: Autores