UNIVERSIDAD INTERNACIONAL DEL ECUADOR - LOJA

ESCUELA DE INFORMÁTICA Y MULTIMEDIA

TESIS DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA Y MULTIMEDIA

ELABORACIÓN DE SOFTWARE DE CONTROL Y EMISIÓN DE PUBLICIDAD PARA UV TELEVISIÓN

CORREA REQUENA ALAN YAMIL

DIRECTOR
ING. ÁLEX VINICIO PADILLA ENCALADA MGS.

Abril 2015
Loja – Ecuador

Yo, Alan Yamil Correa Requena, declaro bajo juramento, que el trabajo aquí descrito es de mi autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedo mis derechos de propiedad intelectual a la Universidad Internacional del ecuador, para que sea publicado y divulgado en internet, según lo establecido en la ley de propiedad intelectual, reglamento y leyes.

Atentamente:

Alan Yamil Correa Requena.

Autor

Yo, Ing. Álex Vinicio Padilla Encalada Mgs., certifico que conozco al autor del presente trabajo siendo él responsable exclusivo tanto de su originalidad y autenticidad, como de su contenido

Atentamente:

Ing. Álex Vinicio Padilla Encalada Mgs.
Director de la tesis

Esta tesis se la dedico a mi Dios quién supo guiarme por el buen camino, darme fuerzas para seguir adelante y no desmayar en los problemas que se presentaban, enseñándome a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento.

A mi familia quienes por ellos soy lo que soy.

Para mis padres por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar. Me han dado todo lo que soy como persona, mis valores, mis principios, mi carácter, mi empeño, mi perseverancia, mi coraje para conseguir mis objetivos.

A mis hermanos por estar siempre presentes, acompañándome para poderme realizar.

A mi novia Lily quien ha sido y es mi motivación, inspiración y felicidad.

A Dios, a mi director de tesis el Ing. Ing. Álex Vinicio Padilla Encalada Mgs., a las personas que colaboraron de una u otra forma para la realización de este trabajo, y especialmente a mis padres por todo su apoyo y la oportunidad de poder estudiar.

Resumen

Con el fin de contribuir al mejoramiento de las actividades involucradas en el proceso de organización y emisión de la publicidad se ha elaborado un software que permita registrar y agendar spots publicitarios de manera tal que se dé cumplimiento exacto a las órdenes publicitarias o contratos relacionados.

El proyecto se desarrolló bajo la metodología Iconix con la participación activa de los usuarios finales del software.

Con este proyecto se pretende reemplazar el proceso manual de organización y emisión de spots publicitarios, con una alternativa más eficiente que centraliza la administración para que en una única tarea se realicen los pasos que antes tomaban mucho tiempo y esfuerzo, es así que al ingresar al sistema un registro de spot publicitario obtenemos en un solo paso el reporte detallado de emisión, el certificado de transmisión y la generación de la lista de reproducción, gracias a que se enlaza el registro de la base de datos con los archivos de video.

Los usuarios se clasifican en dos grupos para garantizar que solo aquellos que estén autorizados puedan realizar cambios de fondo en la organización de la emisión.

La característica especial del software es un módulo de reproducción que recaba los horarios programados y elabora una lista de reproducción que representa un corte comercial de televisión.

Abstract

In order to improve the activities involved in the advertising process on TV channels, we have developed software which is able to register and schedule advertising spots in such a way that the broadcast order or contract requirements are met.

This project has been developed under Iconix methodology and featured the active participation of the final users of the system.

This project represents a change in the process of advertising broadcasting, improving the process efficiency. The administration has been centralized in order to achieve that the time consuming steps are performed in a single task, thus, when a spot record is registered, we get the broadcast report, the broadcast certificate and the playlist done, thanks to the link between the database records and the video files.

Users are divided in two groups to assure that only the authorized users can alter the major parameters in the broadcast process.

The software's special feature is a play out module that collects the existing schedules to generate a play list that is the equivalent of a TV commercial break.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo Análisis Preliminar

1.1. Introducción.

En el mundo moderno son pocas las actividades que se realizan sin la ayuda de equipos de computación y software especializado o de consumo masivo; los canales de televisión no son la excepción: en las etapas de la producción televisiva intervienen distintos equipos como video cámaras, equipos sofisticados de grabación y recepción satelital, estaciones de edición digital y post producción.

Para estas etapas de preparación de contenidos multimedia, el mercado ofrece una amplia variedad de productos tanto de hardware como de software, entre los cuales resaltan Apple Computer y Adobe Systems, los cuales prácticamente se han convertido en el estándar en la producción de video digital a costos asequibles.

Sin embargo para la etapa final que es la emisión de contenidos (broadcasting) no se ha logrado establecer una tendencia clara. Aunque existe software profesional orientado al broadcasting, éste normalmente está enlazado a una determinada línea de hardware y los costos son bastante elevados (desde $8000 hasta $ 130000 si incluyen hardware). También hay alternativas más económicas alrededor de los $700 - $1000 que satisfacen los requerimientos de algunas estaciones de televisión de tamaño pequeño, pero aun así no logran mejoras sustanciales sobre los procesos manuales dado que las interfaces y los flujos de trabajo deben acondicionarse a un rango demasiado amplio de requerimientos. (bhphotovideo, 2006)

Para la elección de paquetes informáticos en empresas, la respuesta más acertada es casi siempre un software a la medida, aunque existan conocidas excepciones como el software de ofimática.

El presente proyecto pretende convertirse en una alternativa para un sector marcado del broadcasting, que es la emisión y control de publicidad. Tomando como premisa mantener una operación sencilla y ser capaz de operar con elementos de hardware de distintas marcas y características, el software desarrollado en este proyecto resulta una opción viable para los canales locales y nacionales que no pueden acceder a costosas y complicadas soluciones. Particularmente se desarrollará para el flujo de trabajo de UV Televisión.

1.1.1. Exposición del problema.

El proceso de organización y emisión de la publicidad en UV Televisión al momento de recabar los requerimientos se realizaba de la siguiente forma:

El personal administrativo se encarga de realizar la contratación y archivar en forma impresa las órdenes de emisión y contratos.

En una computadora ubicada en el estudio central (denominada SERVER por el nombre de equipo configurado en Windows) se ha creado una carpeta para cada día de la semana, estas carpetas contienen proyectos de Adobe Premiere 5.5, los cuales corresponden cada uno a un programa que se emite en ese día, los proyectos contienen los clips de video que deben emitirse en el programa correspondiente; de existir más de un corte comercial, se

separa los bloques mediante un espacio vacío de duración no definida pero suficiente para hacer notar al operario (operario = switcher de ahora en adelante) que se trata de otro espacio publicitario.

Figura 1.1: Proyecto de Adobe Premiere

Fuente: UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Para emitir el corte publicitario, el operario se ubica en el timeline del Adobe Premiere en el sitio exacto donde quiera empezar la reproducción y presiona el botón de reproducción o el atajo que es la barra espaciadora. El video se reproduce a través de la salida de video del SERVER.

Ejemplo: se realiza un contrato con el anunciante X para difundir el producto XYZ en los horarios especificados en el contrato y por el periodo definido en el mismo documento; el encargado traslada el archivo de video al SERVER, luego, en un espacio de tiempo en el cual no se está emitiendo publicidad, el encargado abre uno a uno los proyectos de Adobe Premiere y en los programas que corresponda va insertando el clip XYZ.

Cuando el contrato finaliza, el encargado debe dirigirse otra vez al SERVER, esperar que no se esté emitiendo publicidad, abrir uno a uno los proyectos de Adobe Premiere, ubicar el clip XYZ y eliminarlo del proyecto.

En este proceso se puede identificar algunos puntos débiles:

- Hay un registro de la publicidad, pero es manual y la información no está a la mano de forma inmediata
- Los reportes para los clientes son elaborados como certificados en forma manual
- El cumplimiento de la emisión depende del encargado y de que éste haya modificado correctamente todos los proyectos de Adobe Premiere
- La modificación de los proyectos no se hace en tiempo real y tiene que esperar a que la emisión se detenga para poder hacerla.
- Para insertar un spot en la programación se debe modificar varios proyectos, lo cual se convierte en una tarea repetitiva y propensa a olvidos y equivocaciones.
- Para comenzar la emisión, el switcher debe colocar el cursor del timeline en el clip que corresponda emitir. Esta tarea puede resultar a veces confusa

porque no hay una delimitación clara en el proyecto de Premiere que indique cuál es el corte que se va a emitir, además con este método no se puede indicar al operario la hora en que debe empezar el corte.

- No es posible actualizar los proyectos de Premiere en forma remota, tiene que ser localmente porque las rutas de los clips tienen direcciones absolutas y porque los proyectos para que puedan utilizar la tarjeta de video de salida al aire tienen que ser creados y modificados con el hardware y los plugins instalados en el SERVER.

1.2. ICONIX

Consiste en un lenguaje de modelado y un proceso. El lenguaje de modelado es la notación gráfica (incluye diferentes tipos de diagramas), el proceso define quién debe hacer qué, cuándo y cómo alcanzar un objetivo. (Rosenberg, 2011)

ICONIX es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

Presenta claramente las actividades de cada etapa y exhibe una secuencia de pasos que deben ser seguidos.

Está entre la complejidad del RUP (Rational Unified Processes) y la simplicidad de XP (Extreme Programming).

1.2.1. Características

- Iterativo e incremental: varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.
- Trazabilidad: cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes "artefactos de software" producidos.
- Dinámica del UML: La metodología ofrece un uso "dinámico" del UML por que utiliza algunos diagramas del UML, sin exigir la utilización de todos, como en el caso de RUP.

1.2.2. Fases

1.2.2.1. Análisis de Requisitos

Se realiza un relevamiento de todos los requisitos que en principio deberían ser parte del sistema. Se debe capturar información sobre lo que les gusta y lo que les desagrada a los usuarios. Comprende:

a. Modelo de Dominio.- Con los requisitos se construye el diagrama de clases, que representa el modelo estático del sistema.
b. Prototipación Rápida.- Se usa para simular el diseño del sistema. Se espera que los usuarios lo evalúen como si fuera el sistema final. Los cambios al prototipo son planificados con los usuarios antes de llevarlos a cabo. El

proceso se repite y finaliza cuando los usuarios y analistas están de acuerdo en que el sistema ha evolucionado lo suficiente como para incluir todas las características necesarias o cuando es evidente que no se obtendrá mayor beneficio con una iteración adicional.

c. Modelo de Casos de Uso.- El modelo de los casos de uso comprende los actores, el sistema y los propios casos de uso. Los casos de uso permiten a los usuarios estructurar y articular sus deseos; les obligan a definir la manera como querrían interactuar con el sistema, a precisar qué informaciones quieren intercambiar y a describir lo que debe hacerse para obtener el resultado esperado.

1.2.2.2. Análisis y Diseño Preliminar

Descripción de Casos de Uso.- Los Casos de Uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista de un usuario; permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.

1.2.2.3. Diseño

Diagrama de Secuencia.- Es el núcleo del modelo dinámico y muestra todos los cursos alternos que pueden tomar los casos de uso. Especifica el comportamiento. La representación se concentra sobre la expresión de las interacciones. Se componen de 4 elementos que son: el curso de acción, los objetos, los mensajes y los métodos.

1.2.2.4. Implementación

Escribir / Generar el Código.- La importancia de la interactividad, accesibilidad y navegación en el software harán que el usuario se sienta seguro y cómodo al hacer uso de la aplicación sin inconvenientes.

1.2.3. Otros factores de consideración

Se deben considerar factores como:

- La Reusabilidad: que es la posibilidad de hacer uso de los componentes en diferentes aplicaciones.
- La Extensibilidad: que consiste en modificar con facilidad el software.
- La Confiabilidad: realización de sistemas descartando las posibilidades de error.
- Realizar pruebas. Test de casos, datos y resultados. Test de integración con los usuarios para verificar la aceptación de los resultados.

Lo original de la metodología es la definición de un proceso ágil para obtener la especificación de requerimientos y modelar el comportamiento de sistemas, utilizando el lenguaje de modelamiento unificado (UML).

Es una alternativa que se acopla al desarrollo del sistema en cuestión, que favorece la participación de los usuarios finales y la documentación de todo el proceso.

La metodología ICONIX resuelve el 80% de los desarrollos de software utilizando sólo un 20% de los modelos definidos en UML. No descarta la utilización de ninguno de los modelos, en los casos en que sea necesario, sino que define un conjunto mínimo de modelos y un proceso dinámico de desarrollo, utilizable en la mayoría de los casos.

2. Capítulo Análisis de Requisitos

2.1. Análisis de requerimientos

En aplicaciones de software y hardware, los requerimientos de software son las características que debe tener el software instalado en una computadora para poder soportar y/o ejecutar una aplicación o un dispositivo específico.

Los requerimientos de software pueden ser:

- Requisitos de sistema operativo.
- Requisitos de aplicaciones específicas instaladas.
- Requisitos de ciertas aplicaciones no instaladas en el mismo sistema.
- Requisitos de determinadas configuraciones en el sistema operativo o en ciertas aplicaciones.

2.1.1. Recopilación de información

El Tesista ha sido autorizado (ver el Anexo 2) a revisar los procesos involucrados en la emisión de la publicidad en UV Televisión, desde el ingreso de nuevos contratos publicitarios hasta la emisión final o broadcasting.

- Se analizaron varias órdenes de emisión y algunos contratos publicitarios.
- Se realizó un análisis del proceso actual de emisión.

- Se recabó información y sugerencias del personal que labora en UV Televisión, en forma verbal.

2.1.2. Resumen de la recopilación de información

Una vez que se recopiló la información detallada anteriormente, se especificó los procesos que se llevan a cabo en las áreas de Secretaría y Control Master, tal como se describen a continuación:

Tabla 2.1 Proceso de contratación

Proceso de contratación	
Registrar las órdenes de emisión	El usuario requiere ingresar y actualizar las órdenes de emisión y/o los contratos publicitarios, no de forma textual sino más bien como un extracto de la información requerida como: periodo del contrato, cliente, número de ordeno de contrato, detalle de horarios.
Organizar la distribución de espacios publicitarios	El usuario necesita conocer los espacios disponibles para ubicar nuevos spots publicitarios. Esto es útil para coordinar con el departamento de ventas.
Reportar cumplimiento	El usuario necesita un reporte impreso o digital del detalle de emisión para presentar al cliente.

Tabla 2.2 Proceso de emisión de la publicidad

Proceso de emisión	
Emitir la publicidad	El usuario del control master requiere una forma simple de emitir los spots publicitarios.
Evitar errores	El usuario del control master necesita que se emita lo estrictamente programado según la contratación ya que existen multas y sanciones por incumplimiento de contratos y por emitir publicidad no autorizada o no contratada.
Sincronizar con la emisión de programas	El usuario necesita conocer la duración de los cortes comerciales para que el director de emisión pueda coordinar los programas en vivo y saber si tiene que acelerar o ralentizar los guiones para que el siguiente programa comience a tiempo.
Automatizar la emisión	El usuario necesita despreocuparse de los ingresos de última hora, de las suspensiones no comunicadas o del cambio de pautaje, que cualquier cambio que se realice, se refleje en tiempo real.

Fuente: Personal UV TELEVISIÓN.
Autor: Alan Yamil Correa Requena.

2.2. Identificación de las clases, objetos, procesos y conceptos

De acuerdo a los resultados obtenidos anteriormente, se decide dividir el proyecto en 5 módulos: SPOTS, PROGRAMAS, CORTES, PARRILLA, EMISIÓN.

Con la premisa mencionada se identifican los conceptos a utilizarse en los diferentes módulos de manera compartida en el proyecto.

Tabla 2.3 Conceptos

Conceptos	
Nombre	**Descripción**
Spot	Puede referirse a un clip de video, a un registro de base de datos que represente a un spot o a una unidad cuando se contabilizan detalles de emisión.
Orden de emisión	Hace alusión a un documento que contiene los detalles de emisión. Este documento puede ser un contrato o una orden de emisión enviada por una agencia de publicidad.
Parrilla	Se refiere a una grilla que detalla los cortes comerciales de un periodo determinado y su contenido.
Control Master	Es el área del canal donde se manejan los equipos de broadcasting o emisión, se coordina la misma y se monitorea toda la actividad.
Corte comercial	O simplemente "corte", es un espacio publicitario que se emite de forma periódica.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

2.3. Elaborar un modelo de dominio

Un modelo de dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software, este muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. (Larman, 2003)

Un modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física.

Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema.

Es posible capturar un mayor grado de detalle en uno de estos modelos; corresponde al analista decidir el nivel de detalle que va a ser necesario y hasta donde llegar a modelar, el objetivo es capturar lo necesario para comprender donde va a funcionar el sistema que estamos diseñando y esto demanda una cantidad distinta de detalles cada vez.

El modelo de dominio en la metodología ICONIX es el punto de partida para el diseño del sistema. Esto es así ya que cuando se realiza la programación orientada a objetos, se supone que el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de domino constituye una primera versión del sistema.

2.3.1. ¿Cómo se elabora el modelo de dominio?

El proceso para su elaboración tiene tres pasos:

1. Identificar las clases conceptuales.
2. Dibujarlas en un Diagrama de Clases.
3. Añadir relaciones y atributos.

2.3.1.1. Identificar las Clases Conceptuales

Una "clase conceptual" es cualquier cosa que pertenezca al dominio, por ejemplo, personas, máquinas, lugares, etc. también elementos intangibles como conceptos (venta, permiso, perfil, etc.).

La mejor forma de identificar estas clases es escuchando a los Expertos de Negocio que son las personas involucradas directamente en el proceso y conocen qué es lo que se quiere conseguir (operarios, directivos, asistentes). Bastará una reunión con ellos en la que se comente todo el dominio de la aplicación para identificar las clases conceptuales. Se debe considerar que las clases suelen ser los sustantivos utilizados por los expertos al describir el dominio. También se puede reutilizar modelos existentes o recurrir a listas predefinidas, pero deben utilizarse sólo como medios complementarios a la escucha de los expertos.

2.3.1.2. Dibujar un Diagrama de Clases

El Modelo de Dominio queda recogido en un diagrama (parecido al Diagrama de Clases pero más simplificado).

Se representa cada clase por su nombre, dentro de un recuadro.

Nota: Un error habitual es confundir las clases conceptuales con clases de software. Puede que finalmente, al realizar el diseño de clases, haya clases de software con el mismo nombre que las clases conceptuales del modelo de dominio. Pero en ningún caso son las mismas. Y aunque las diferencias puedan ser sutiles, si se aplican atributos y características propias del software a las clases conceptuales se estarán incluyendo restricciones innecesarias para el diseño de clases.

El objetivo es comprender, no documentar, así que el Modelo de Dominio sólo recogerá las clases relevantes para el problema que se requería entender en ese momento.

2.3.1.3. Añadir Relaciones y Atributos

Entre las clases existen relaciones que pueden identificarse como los verbos utilizados por los expertos de dominio.

En el diagrama, cada relación queda representada por una línea que une las clases relacionadas (no sus atributos).

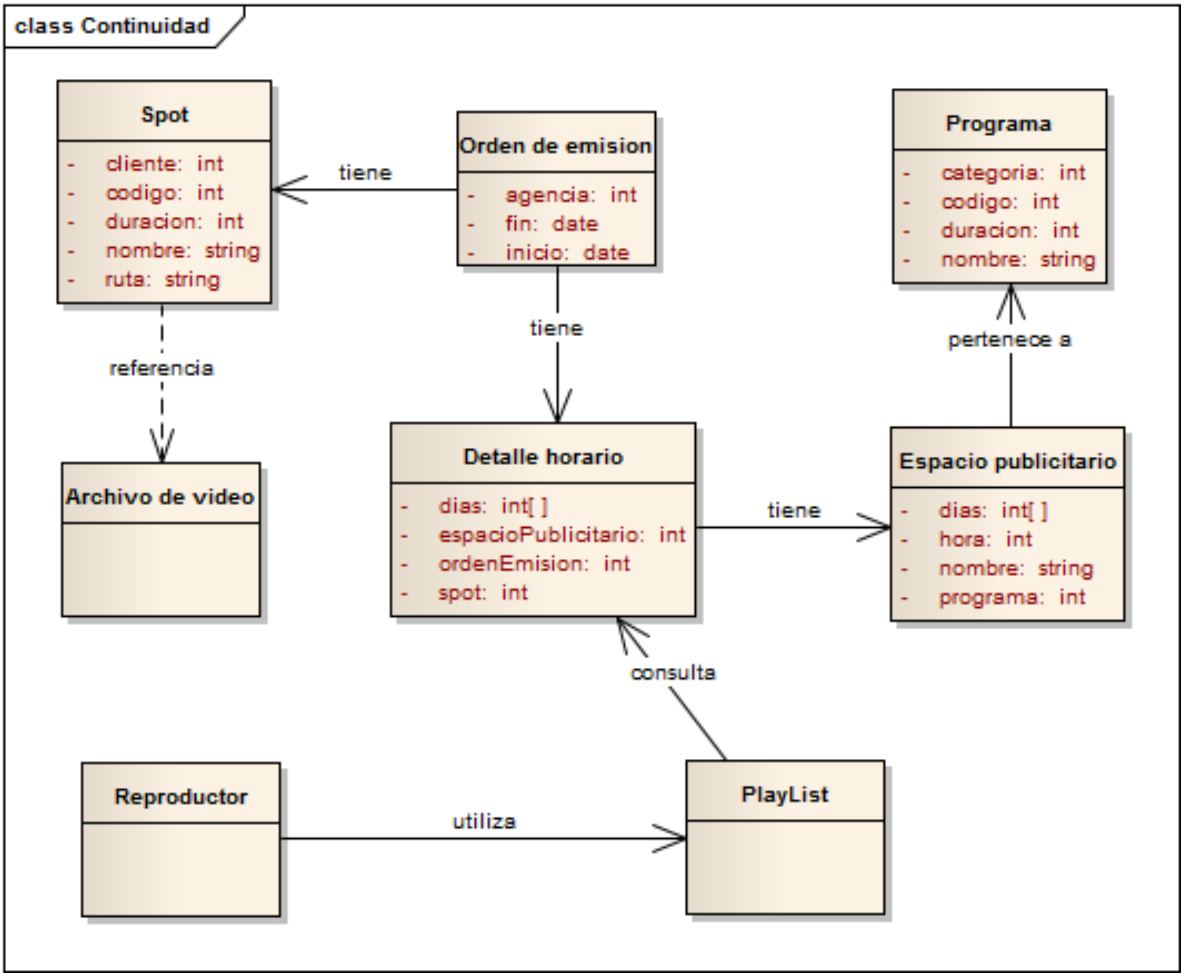
Nota: Las relaciones del Modelo de Dominio no son relaciones entre clases, son relaciones entre instancias de las clases. Es decir, el Modelo de Dominio no representa relaciones de herencia, uso, etc., sino relaciones entre instancias de estas clases, que pueden ser "es una parte de", "es resultado de", "es una descripción de", "utiliza a", "pertenece a", etc.

Cada relación tiene también una multiplicidad (si es 1 a 1, n a 1, 0 ó 1, etc.).La multiplicidad de cada relación se representa sobre su línea, junto a la clase correspondiente.

Nota: Las clases conceptuales no tienen claves externas. La mejor forma de expresar estas situaciones es mediante una relación entre las clases. Sólo se representa las relaciones y los atributos que sean relevantes para el problema que queremos entender en ese momento.

2.3.2. Modelo de dominio

Figura 2.1: Modelo de dominio



Autor: Alan Yamil Correa Requena.

2.4. Documento de requerimientos

Se elaboró el documento de requerimientos en base al análisis previo realizado, la recolección de documentación y el conocimiento pleno del dominio del problema.

Tabla 2.4 Requerimientos de Procesos.

REQUERIMIENTOS DE PROCESOS	
CÓDIGO	**DESCRIPCIÓN**
RP01	*Módulo Clientes:* Este módulo sirve para **gestionar** los clientes que se relacionan con los spots publicitarios.
RP02	*Módulo Spots:* Este módulo permite **gestionar**los datos de los spots, relacionarlos a un archivo de video y mostrar una lista filtrada de todos los spots.
RP03	*Módulo Agencias*: Este módulo permite **gestionar** los datos de las agencias de publicidad que se relacionan con las órdenes de emisión.
RP04	*Módulo Ordenes de Emisión: Éste módulo permite **gestionar** las órdenes de emisión y los detalles de los horarios de transmisión de los spots*.
RP05	

	*Módulo Parrilla de Programación: Éste módulo permite **generar**la parrilla de programación diaria a partir de los detalles de las órdenes de emisión*.
RP06	*Módulo Cortes Comerciales:* Éste módulo permite **gestionar** los cortes comerciales en los que se difundirán los spots.
RP07	*Módulo Reproducción: É*ste módulo sirve para **generar** una lista de reproducción de los spots programados para cada corte comercial y reproducirlos a través del hardware de video.

Fuente: Personal UV TELEVISIÓN
Autor: Alan Yamil Correa Requena.

Tabla 2.5 Requerimientos de procesos RP01. Módulo clientes

REQUERIMIENTOS DE PROCESOS RP01. MÓDULO CLIENTES	
CÓDIGO	**DESCRIPCIÓN**
RP01.1	Se ingresa, actualiza o elimina clientes.
RP01.2	Al intentar eliminar un cliente que está relacionado a uno o más spots, el sistema presenta un mensaje y cambia su estado a inactivo.
RP01.3	El sistema controla que no se ingresen clientes duplicados.

Fuente: Personal UV TELEVISIÓN
Autor: Alan Yamil Correa Requena.

Tabla 2.6 Requerimientos de procesos RP02. *Módulo spots*

REQUERIMIENTOS DE PROCESOS RP02. *MÓDULO SPOTS*	
CÓDIGO	**DESCRIPCIÓN**
RP02.1	Se ingresa, actualiza o elimina spots. Para crear un spot es necesario elegir un archivo de video compatible, una vez seleccionado se llenan los campos de nombre, duración y ruta del archivo a partir de la información del archivo elegido. El usuario debe especificar otros datos como el tipo de spot y la categoría a la que pertenece.
RP02.2	El sistema controla que no se registren nombres o archivos duplicados.
RP02.3	Se puede especificar un filtrado de spots para mostrar una lista que cumpla con los criterios seleccionados.
RP02.4	Al modificar los datos de un spot, se cambia automáticamente la fecha de modificación.
RP02.5	Al relacionar un spot con una orden de emisión, la fecha de alerta del spot se modifica automáticamente. Esta es la única forma de definir o modificar la fecha de alerta.
RP02.6	Al intentar eliminar un spot que conste en alguna orden de emisión, el sistema presenta un mensaje y cambia su estado a inactivo.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Tabla 2.7 Requerimientos de procesos RP03. Módulo agencias

REQUERIMIENTOS DE PROCESOS RP03. MÓDULO AGENCIAS	
CÓDIGO	**DESCRIPCIÓN**
RP03.1	Se registra los datos de las agencias de publicidad.
RP03.2	El sistema controla que no se ingresen agencias duplicadas.
RP03.3	Al intentar eliminar una agencia que tenga órdenes de emisión relacionadas, el sistema presenta un mensaje y cambia el estado a inactiva. Si la orden de emisión aún está vigente no es posible eliminar o cambiar el estado de la agencia.
RP03.4	Se valida el RUC de las agencias a través del algoritmo provisto por el SRI.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Tabla 2.8 Requerimientos de procesos RP04. Módulo órdenes de emisión

REQUERIMIENTOS DE PROCESOS RP04. MÓDULO ÓRDENES DE EMISIÓN	
CÓDIGO	**DESCRIPCIÓN**
RP04.1	Se ingresa, modifica o elimina órdenes de emisión, los datos son: spot publicitario, fecha de inicio, fecha de fin, agencia a la que pertenece, número de orden y observaciones.
RP04.2	Se ingresa, modifica o elimina detalles de órdenes de emisión, los detalles incluyen corte comercial y día de emisión (que esté dentro de las fechas de inicio y fin de la orden de emisión), esto define un horario de emisión de spot.
RP04.3	Un spot puede tener varias órdenes de emisión.
RP04.4	Pueden existir varias órdenes de emisión con el mismo número, esto porque existen órdenes de emisión que incluyen más de una versión, es decir distintos spots.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Tabla 2.9 Requerimientos de procesos RP05. Módulo parrilla de programación

REQUERIMIENTOS DE PROCESOS RP05. MÓDULO PARRILLA DE PROGRAMACIÓN	
CÓDIGO	**DESCRIPCIÓN**
RP05.1	Se genera la parrilla de programación del día seleccionado.
RP05.2	La información se muestra agrupada y ordenada por cortes comerciales.
RP05.3	Se muestra la duración individual de los spots y la duración acumulada de los cortes.
RP05.4	Se puede manipular el orden de los spots dentro del corte o mover los spots entre cortes comerciales.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Tabla 2.10 Requerimientos de procesos RP06. Módulo cortes comerciales

REQUERIMIENTOS DE PROCESOS RP06. MÓDULO CORTES COMERCIALES	
CÓDIGO	**DESCRIPCIÓN**
RP06.1	Se ingresa, modifica o elimina cortes comerciales. Se requieren datos de hora del corte, frecuencia (días de la semana en los que se emiten), tipo de corte, nombre del corte.
RP06.2	Al intentar eliminar un corte que consta en detalles de órdenes de emisión, el sistema debe presentar un mensaje y presentar una pantalla para migrar los spots a otro corte comercial.
RP06.3	Si se modifica la hora del corte, el sistema debe verificar que no se cruce con otros cortes comerciales en los mismos días de emisión.
RP06.4	No pueden existir dos cortes con la misma hora en el mismo día.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Tabla 2.11 Requerimientos de procesos RP07. Módulo reproductor

REQUERIMIENTOS DE PROCESOS RP07. MÓDULO REPRODUCTOR	
CÓDIGO	**DESCRIPCIÓN**
RP07.1	Se genera una lista de reproducción para cada corte agendado en el día seleccionado.
RP07.2	Se verifica que existan los archivos de video especificados en las rutas de los spots.
RP07.3	Al seleccionar un corte se genera la lista de reproducción para ese corte.
RP07.4	Se proveen controles de transporte (play, pause, stop)
RP07.5	Se puede agregar o quitar spots a la lista de reproducción en cualquier momento para ajustarse a los tiempos de los programas de televisión. Los spots de tipo autopromoción o spots de bonificación pueden quitarse o aumentarse libremente, los spots comerciales presentarán un mensaje de confirmación.

Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena**.**

2.5. Elaboración del Prototipado

El diseño de prototipos es una técnica popular de ingeniería para desarrollar modelos a escala (o simulados) de un producto o sus componentes, cuando se aplica al desarrollo de sistemas informáticos el diseño de prototipos implica la creación de un modelo o modelos operativos de trabajo de un sistema o subsistemas.

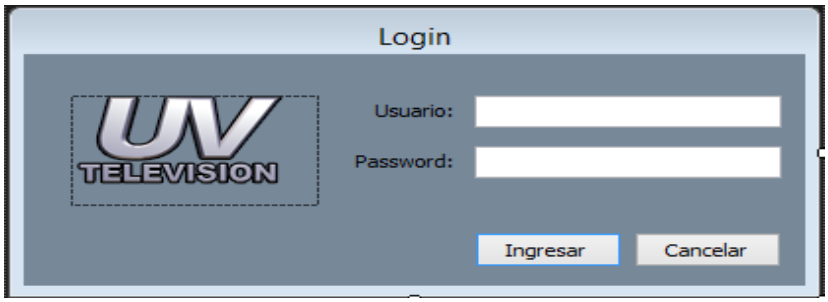
Los prototipos de pantallas proporcionan una manera de obtener las reacciones de los usuarios hacia la cantidad de información presentada sobre la pantalla de visualización, pero también se debe volver a los objetivos para no perder la atención. Su principal propósito es obtener y validar los requerimientos esenciales, manteniendo abiertas las opciones de implementación.

Cuando el prototipo está suficientemente perfeccionado en todos los sentidos requeridos y alcanza las metas para las que fue pensado, el objeto puede empezar a producirse.

2.5.1. Prototipos de formularios principales

2.5.1.1. Login

Figura 2.2: Pantalla Acceso al Sistema

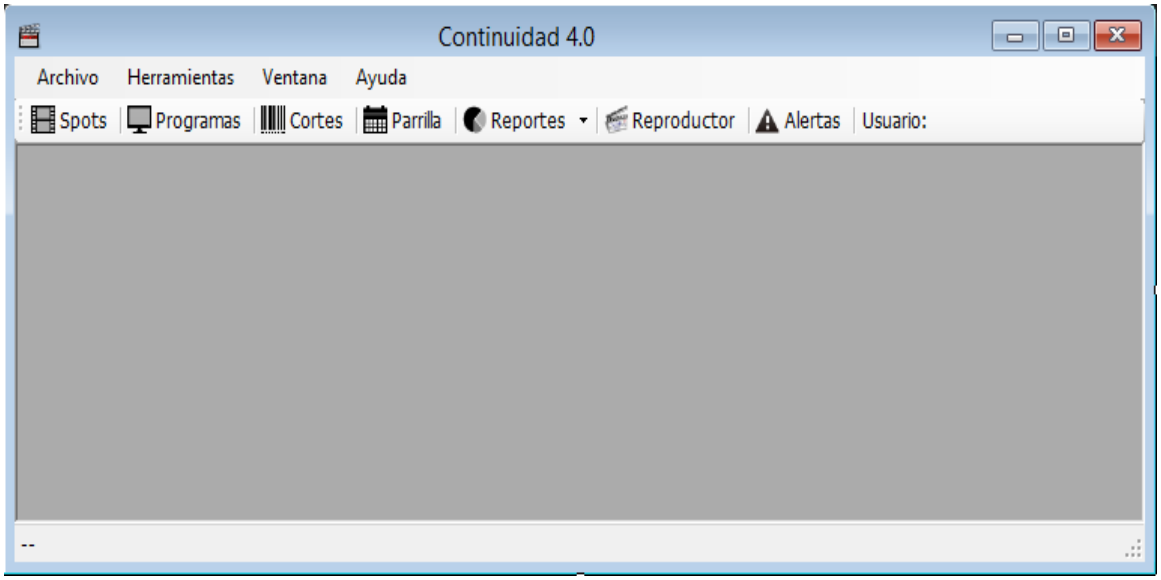


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.1.2. Pantalla principal

Figura 2.3: Pantalla Principal



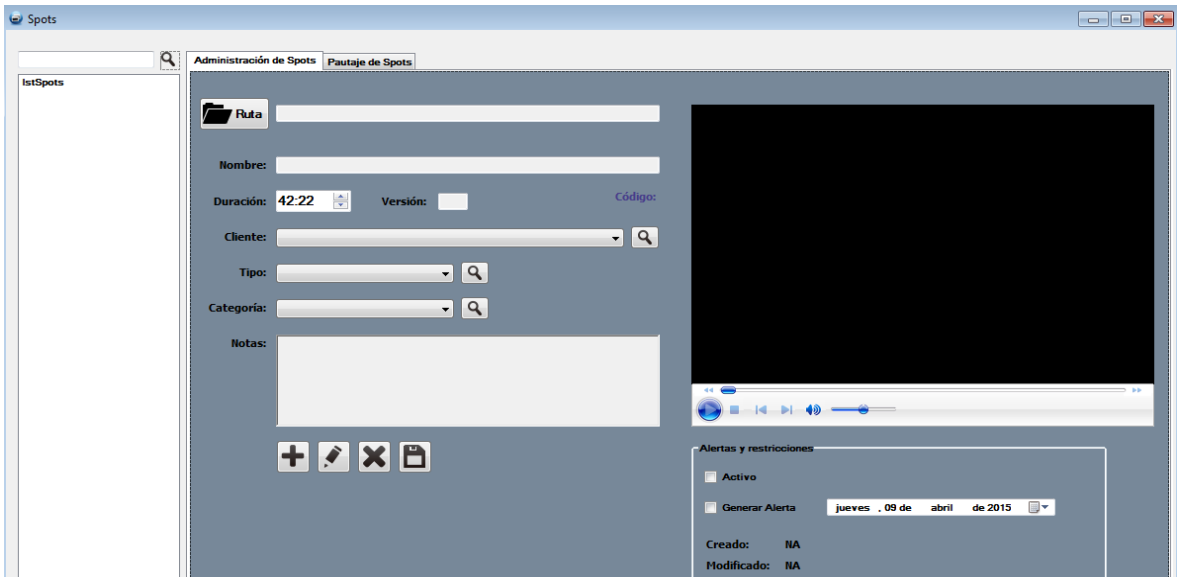
Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2. Prototipos de Módulos

2.5.2.1. Spots

Figura 2.4: Pantalla Spots – Administración

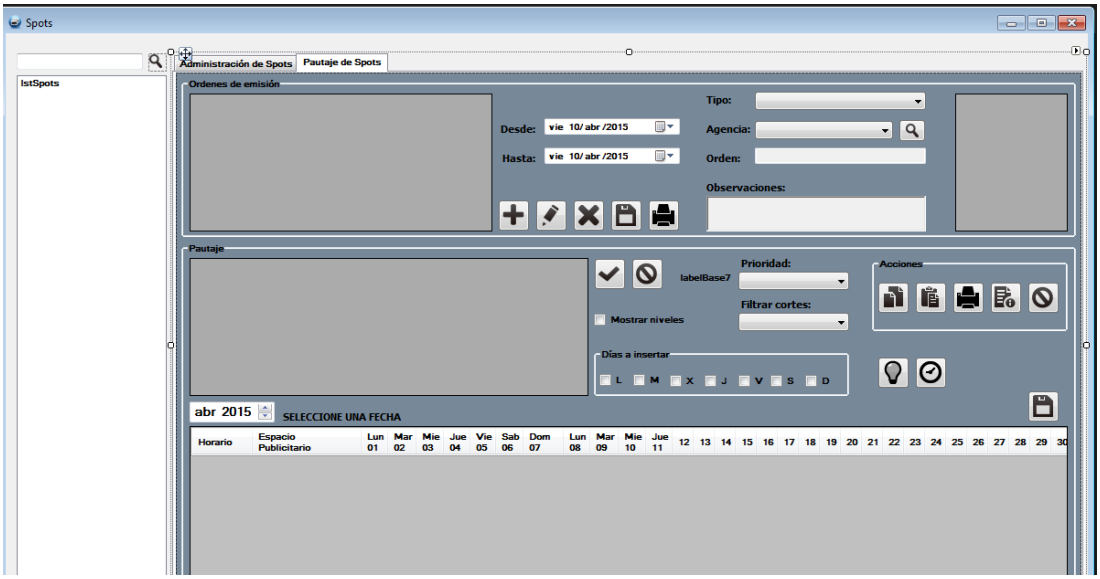


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2.2. Pautaje

Figura 2.5: Pantalla Spots – Pautaje

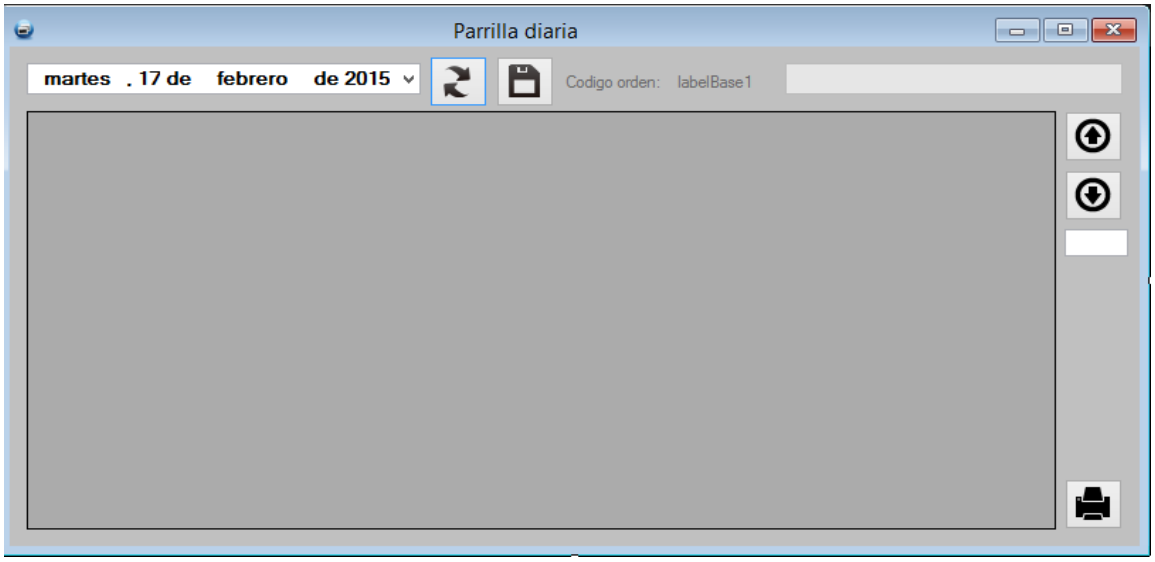


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2.3. Parrilla

Figura 2.6: Pantalla Parrilla

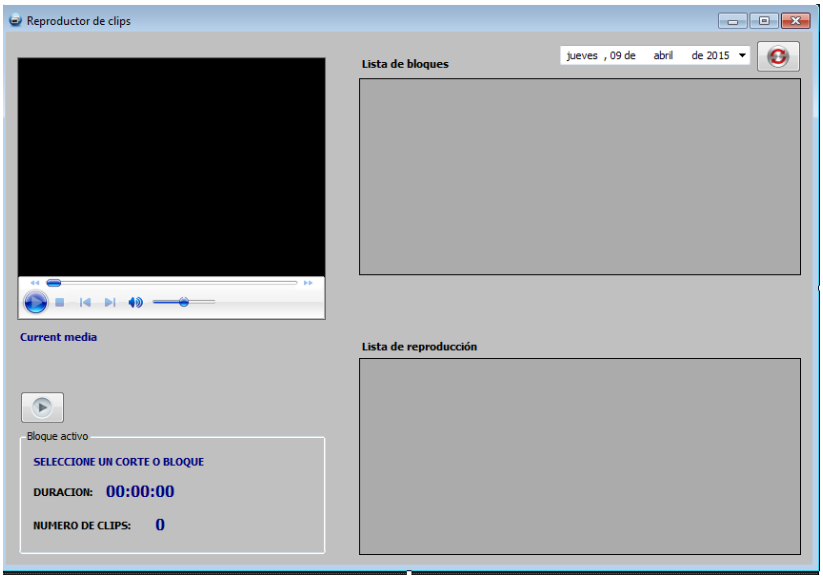


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2.4. Reproductor

Figura 2.7: Pantalla Reproductor

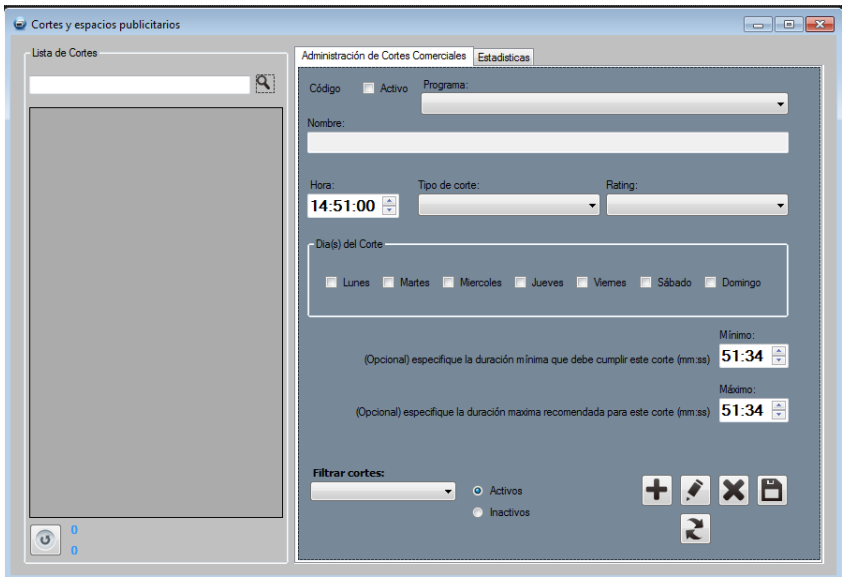


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2.5. Cortes

Figura 2.8: Pantalla Cortes

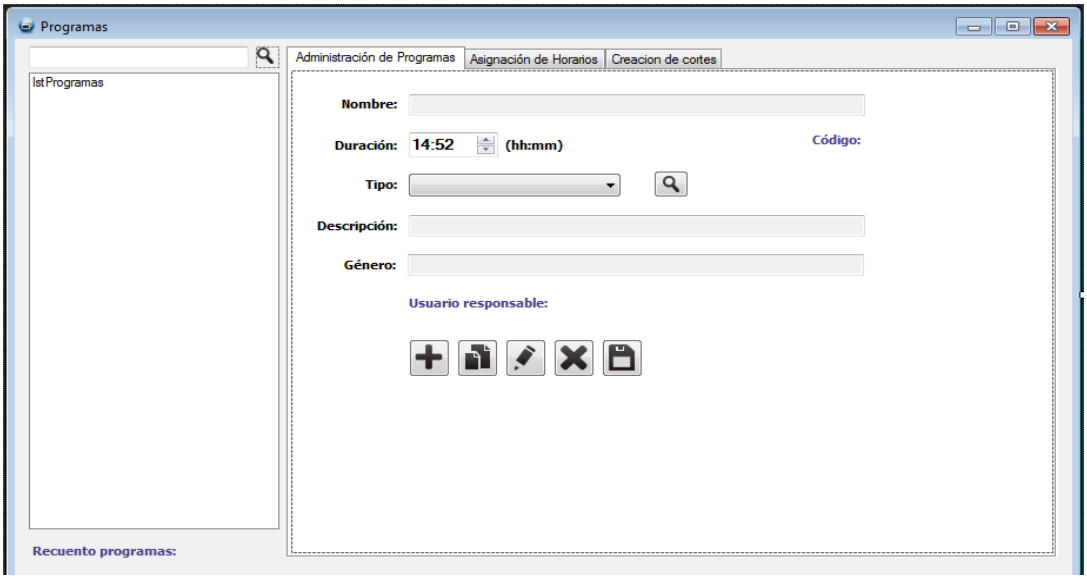


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.2.6. Programas

Figura 2.9: Pantalla Programas



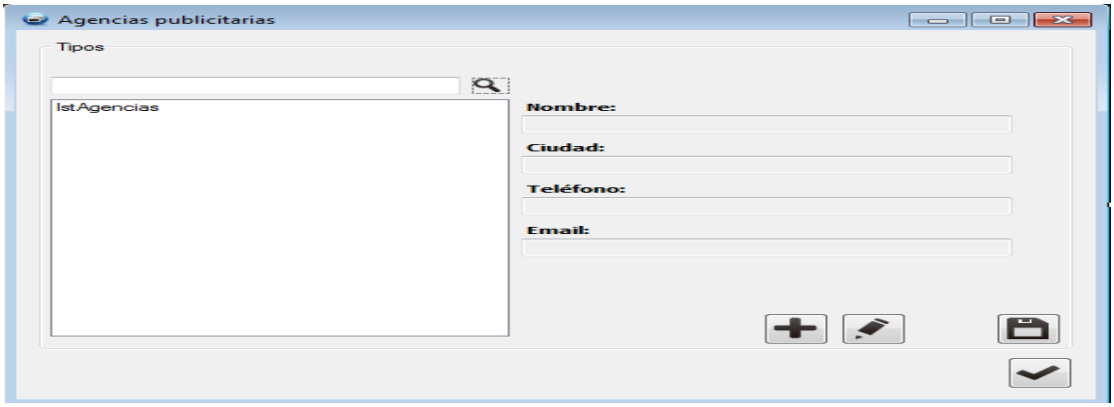
Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.3. Prototipos de Gestión

2.5.3.1. Agencias

Figura 2.10: Pantalla Agencias

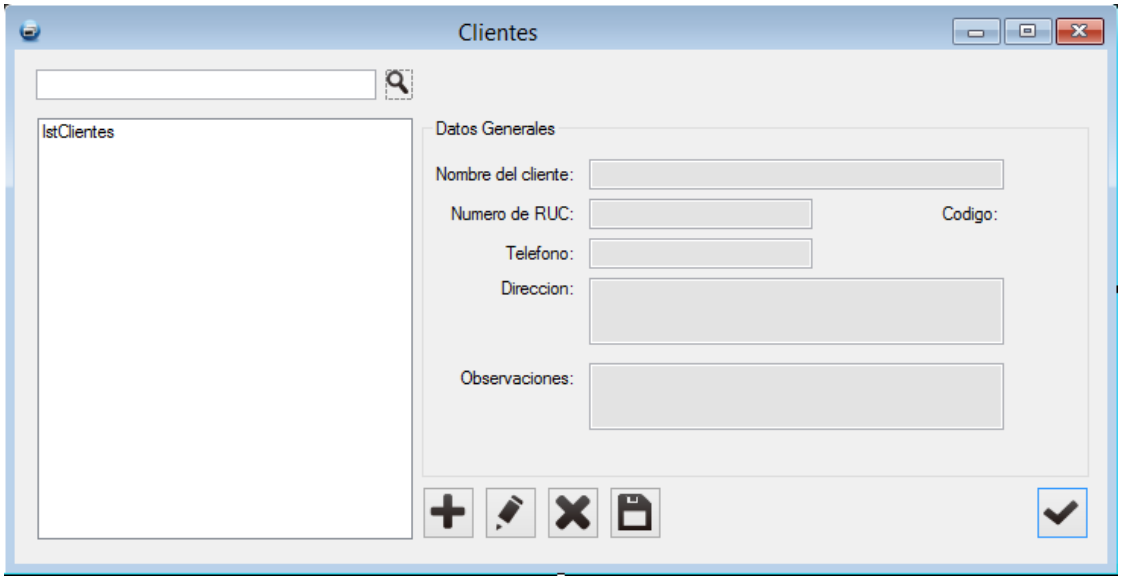


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.3.2. Clientes

Figura 2.11: Pantalla Clientes

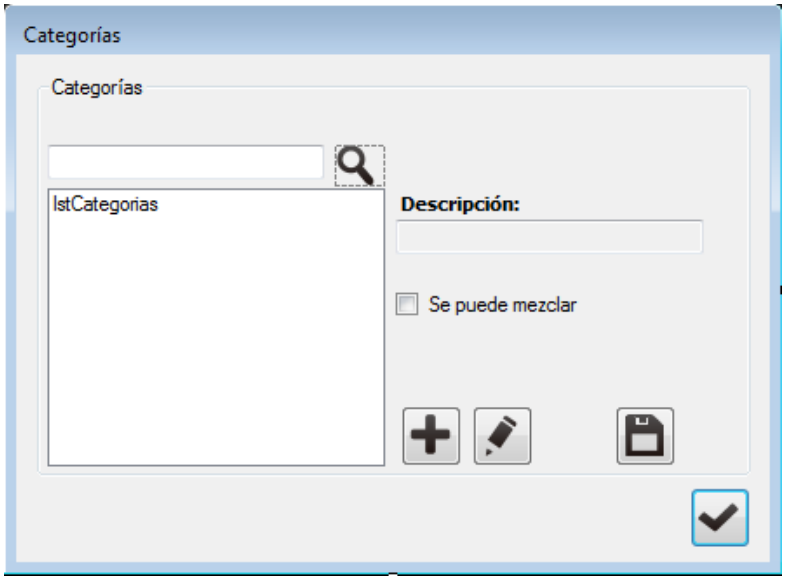


Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.5.3.3. Categorías

Figura 2.12: Pantalla Categorías



Fuente: El Sistema.

Autor: Alan Yamil Correa Requena.

2.6. Identificar los Casos de Uso y los diagramas de Casos de Uso

2.6.1. Introducción

Un caso de uso es una herramienta que sirve para representar la forma como un cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en la cual, los elementos interactúan, a estas acciones se les llama operaciones o Casos de uso.

Los casos de uso se utilizan básicamente en el proceso de modelado de sistemas, partiendo de una percepción o perspectiva que nos plantea el paradigma de la orientación a objetos, y en este caso el análisis y diseño orientados a objetos.

Los casos de uso forman parte del Lenguaje Unificado de Modelado UML por sus siglas en inglés (UnifiedModelingLanguaje) el cual a su vez se compone de muchas otras herramientas, básicamente diagramas como: Diagramas de Clase, Diagramas de Secuencia, Colaboración, Transición de Estados, Diagramas de Actividad, Componentes, Deployment, entre otros. Todas ellas usadas a lo largo de las etapas o ciclo de vida del proceso de desarrollo. (Booch, Grady, Rumbaugh, James, & Jacobson, Ivar, 2005)

2.6.2. Elementos

2.6.2.1. Actor:

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

2.6.2.2. Caso de Uso:

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

2.6.2.3. Relaciones:

- Asociación. Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.
- Dependencia o Instanciación (extend o include). Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.
- Generalización. Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo

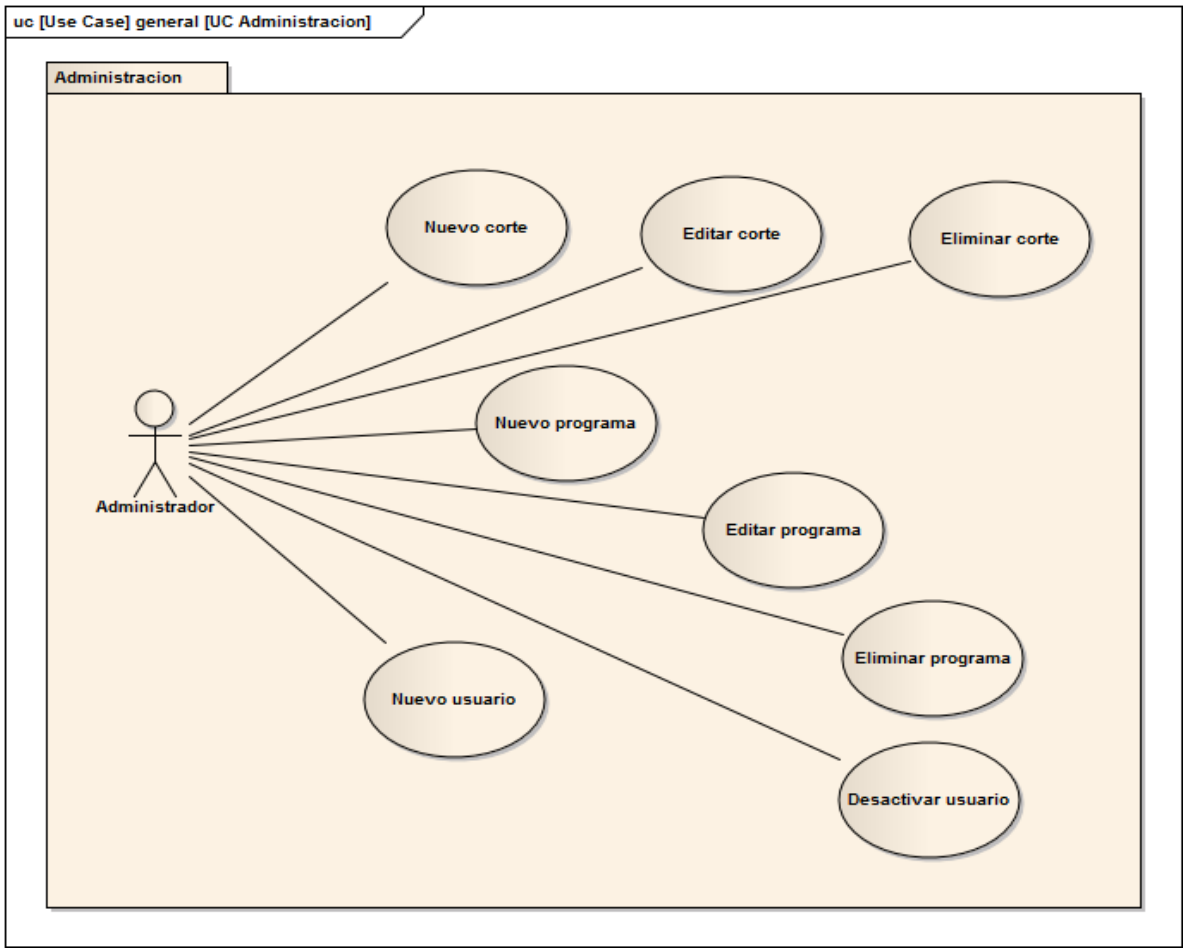
2.6.3. Casos de Uso

Con la información recolectada se identifican los siguientes diagramas de casos de uso, siendo agrupados en paquetes de acuerdo a los módulos a los que corresponden para ilustrar más claramente las relaciones que poseen.

2.6.3.1. Paquetes

Se identifican 3 paquetes: Administración, Gestión y Organización de la Publicidad.

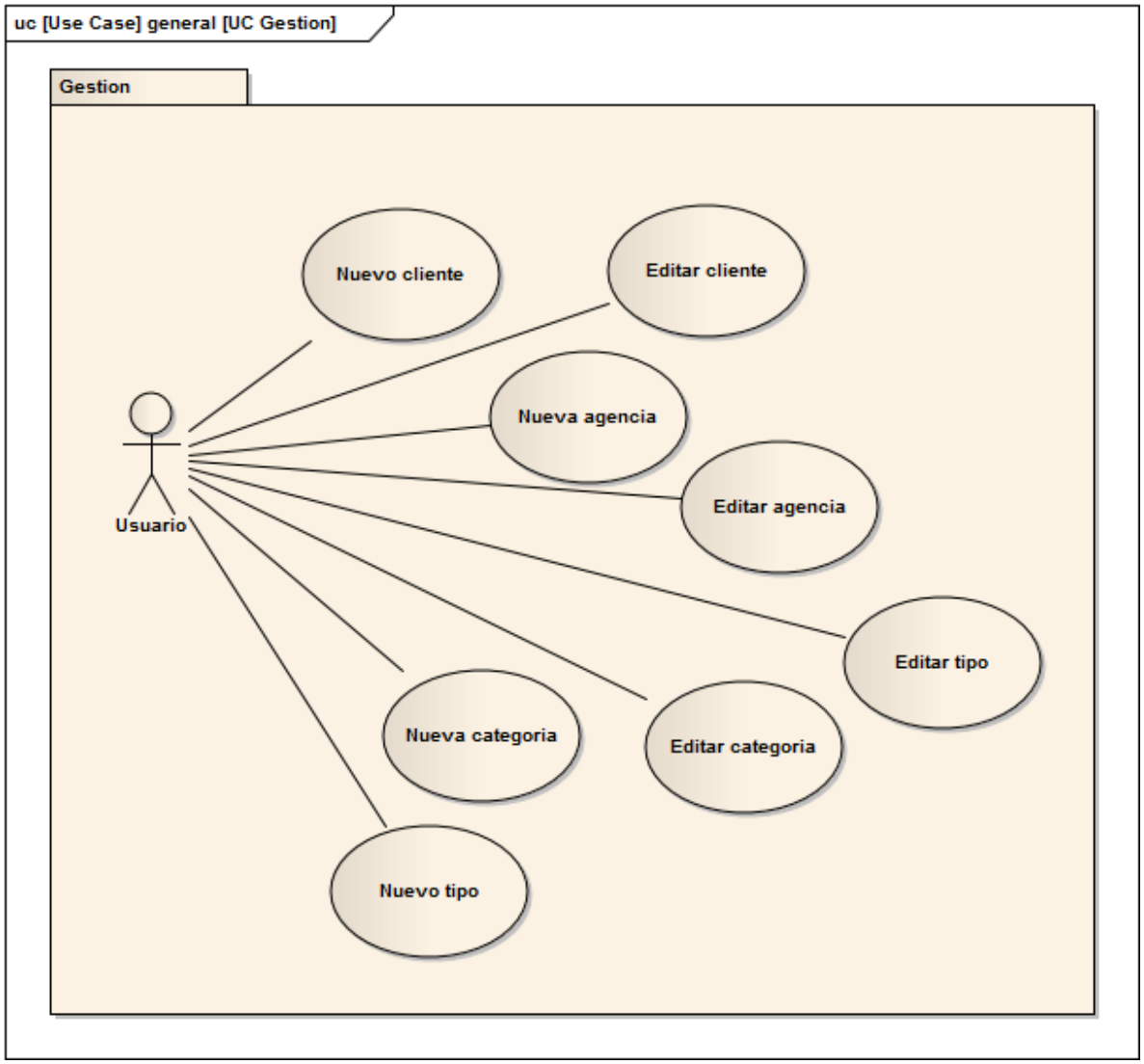
Figura 2.13: Paquete UC Administración



Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

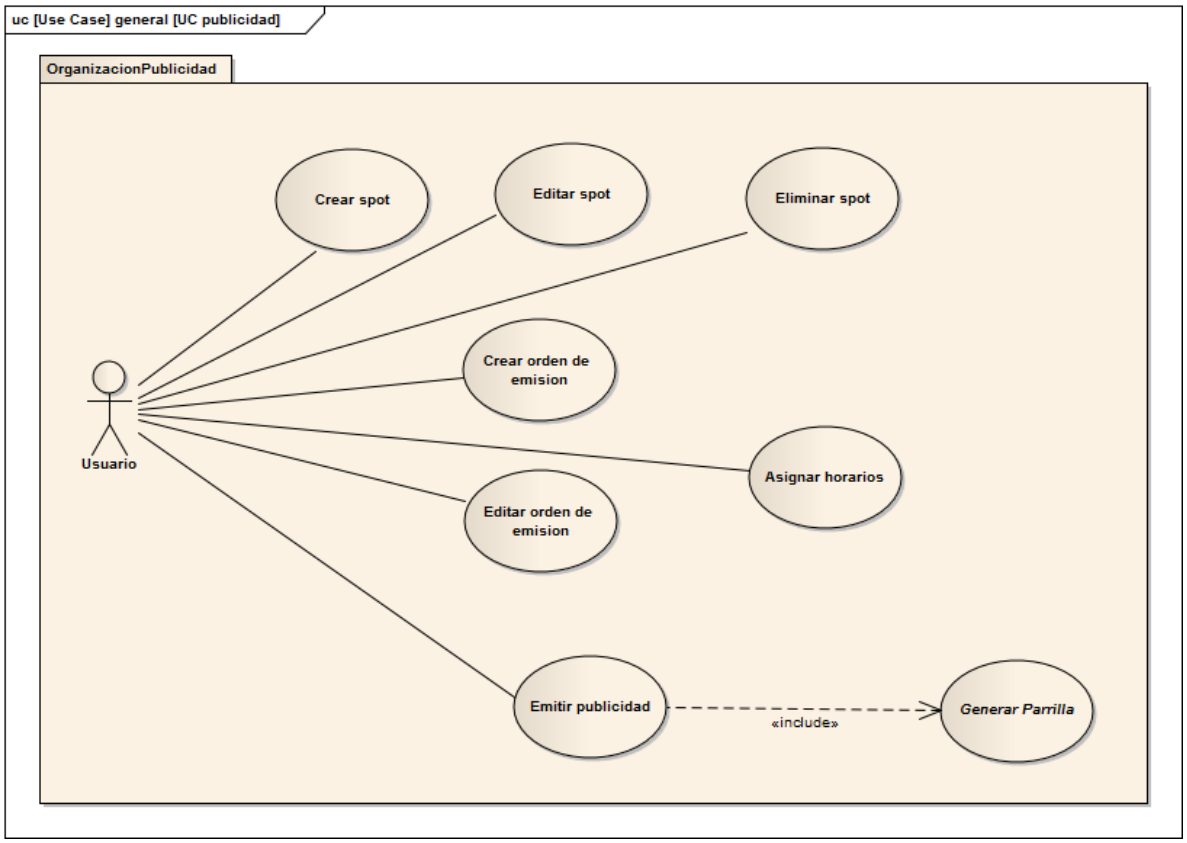
Figura 2.14: Paquete UC Gestión



Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

Figura 2.15: Paquete UC Publicidad



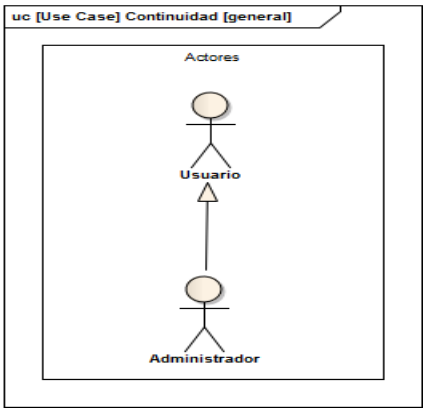
Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

2.6.3.2. Actores

Se identifican los actores, con sus respectivas relaciones, que participan en todos los procesos que serán automatizados.

Figura 2.16: Actores



Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

3. Capítulo Análisis y Diseño

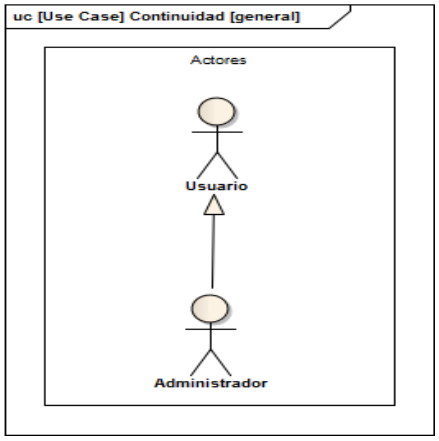
3.1. Análisis y Diseño Preliminar

Se desarrollan descripciones extendidas de los casos de uso que fueron identificados en los diagramas respectivos para establecer el flujo de acciones a desempañarse por cada uno de los actores involucrados y las repuestas y tareas del sistema. Además se elaboran los diagramas de secuencia que clarifican gráficamente el proceso de los casos de uso.

También se elabora detalladamente el diagrama de clases, basado en el modelo de dominio que se obtuvo previamente, con el fin de identificar con exactitud los métodos y atributos de cada una de las clases que se programarán.

3.1.1. Modelo de Actores

Los actores del modelo presentado, dentro del sistema serán identificados por roles de usuario en lugar de referirse a un ente o persona física específica.



Fuente: Personal UV TELEVISIÓN

Autor: Alan Yamil Correa Requena.

3.1.1.1. Administrador

Este actor puede realizar las mismas tareas del Usuario normal (puesto que hereda de éste) y además las tareas concernientes a la administración de elementos importantes dentro del sistema, tales como:

- Crear nuevos Usuarios
- Desactivar Usuarios para impedir su acceso al sistema
- Gestionar Roles de Usuario
- Crear, editar y eliminar cortes comerciales
- Crear y editar programas

3.1.1.2. Usuario

El Usuario representa un rol que permite realizar todas las actividades en el sistema, excepto las del Administrador:

- Gestionar spots
- Gestionar clientes
- Gestionar agencias
- Gestionar tipos
- Gestionar categorías
- Gestionar ordenes de emisión
- Asignar horarios
- Emitir publicidad

3.1.2. Casos de Uso y Diagramas de Secuencia de Administración

3.1.2.1. Nuevo Usuario

El objetivo de este caso de uso es crear un nuevo usuario.

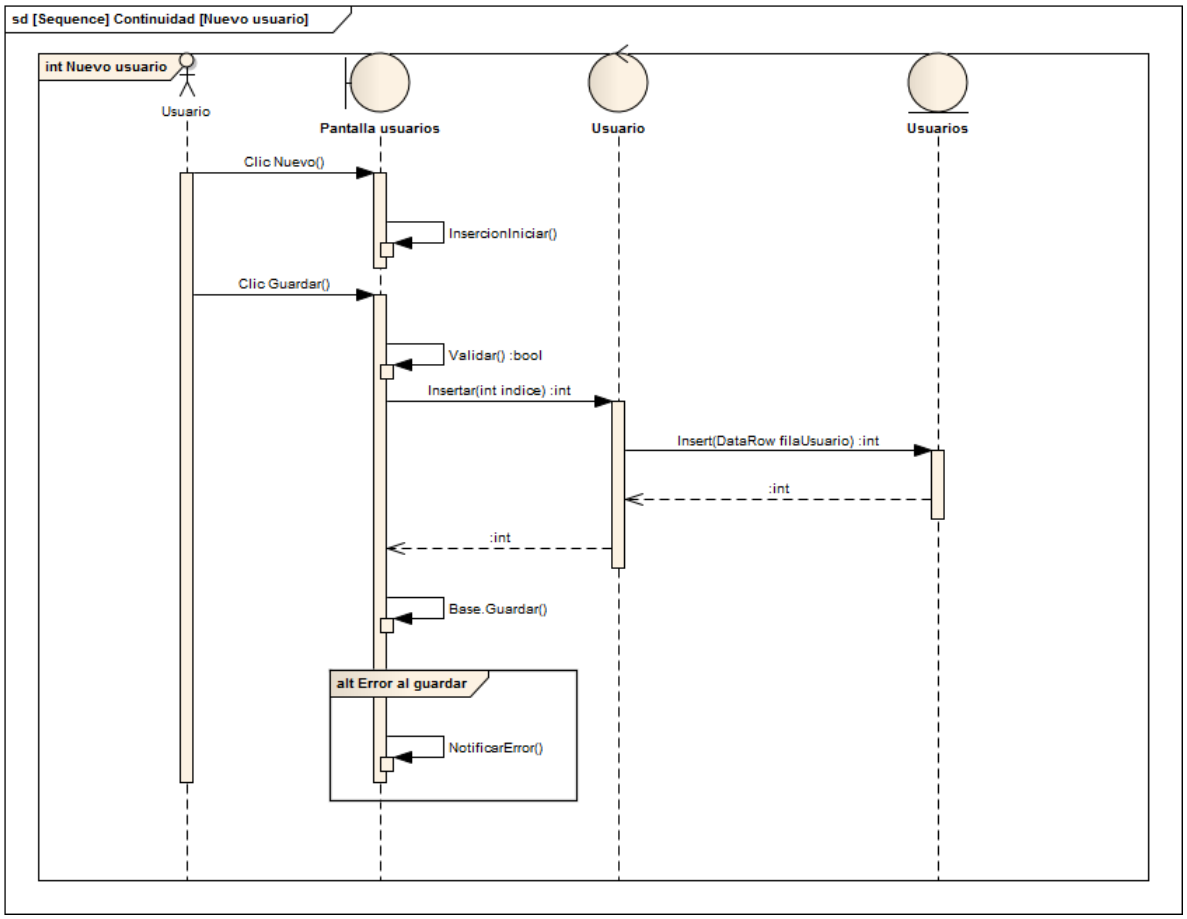
Tabla3.1: Caso de Uso Nuevo Usuario

UC Crear Nuevo Usuario	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Usuarios** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.1: Diagrama de Secuencia Nuevo Usuario



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.2. Desactivar Usuario

El objetivo de este caso de uso es desactivar un usuario para que no pueda ingresar al sistema.

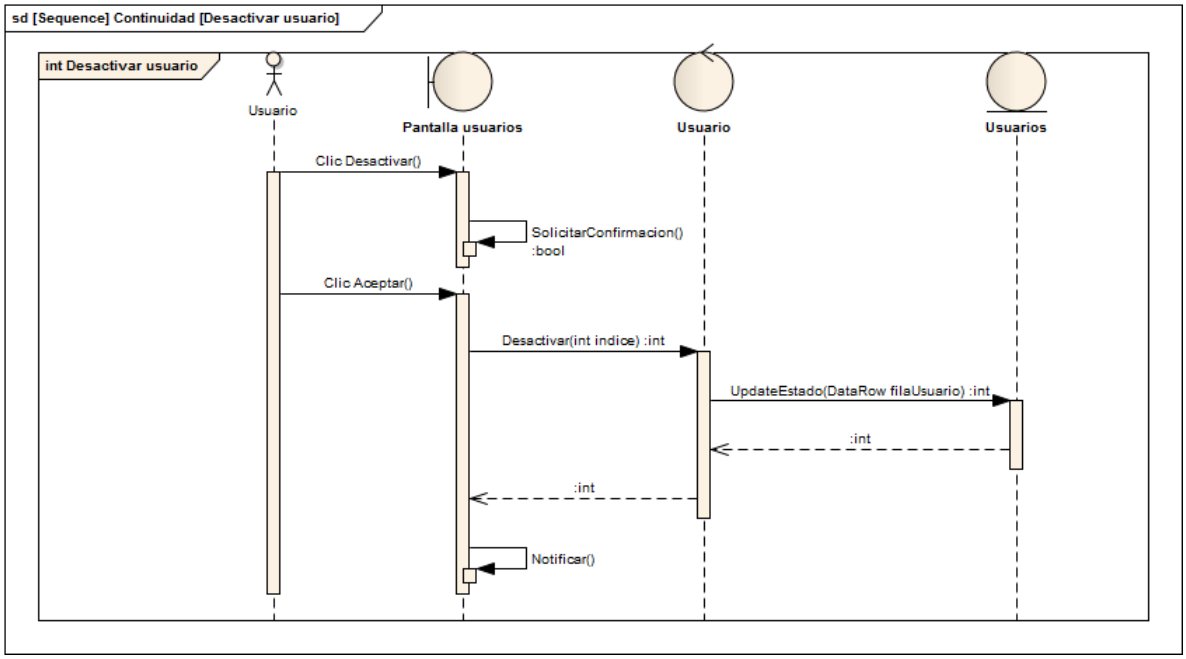
Tabla3.2: Caso de Uso Desactivar Usuario

UC Desactivar Usuario	
•	Curso Normal 1. El Administrador se encuentra en la **pantalla Usuarios** y presiona el botón **Desactivar**. 2. El Sistema solicita confirmación. 3. El Administrador confirma que desea desactivar el usuario seleccionado. 4. El Sistema guarda en la base de datos. 5. El Sistema presenta mensaje.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.2: Diagrama de Secuencia Desactivar Usuario



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.3. Nuevo Corte

El objetivo de este caso de uso es crear un nuevo corte comercial.

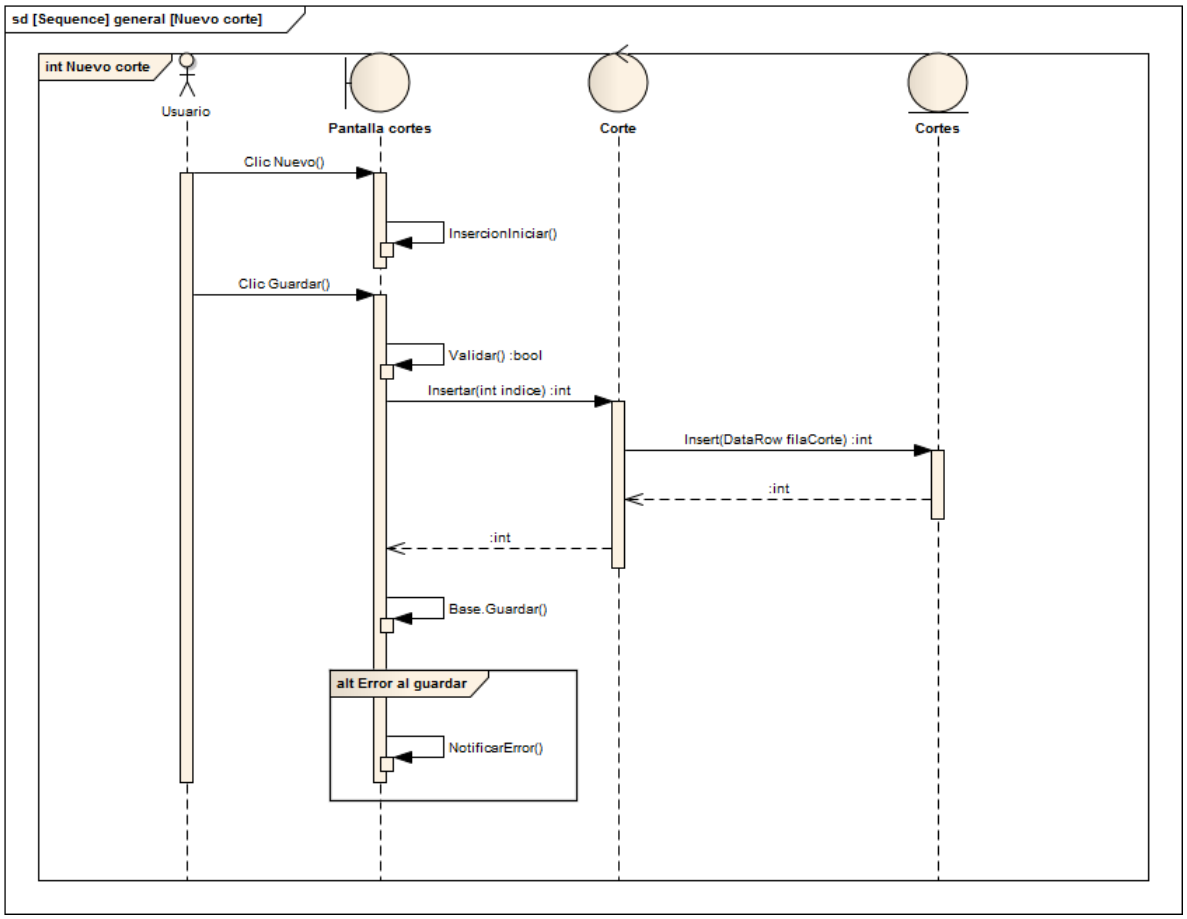
Tabla3.3: Caso de Uso Nuevo Corte

UC Crear Nuevo Corte	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Cortes** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.3: Diagrama de Secuencia Nuevo Corte



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.4. Nuevo Programa

El objetivo de este caso de uso es crear un nuevo corte comercial.

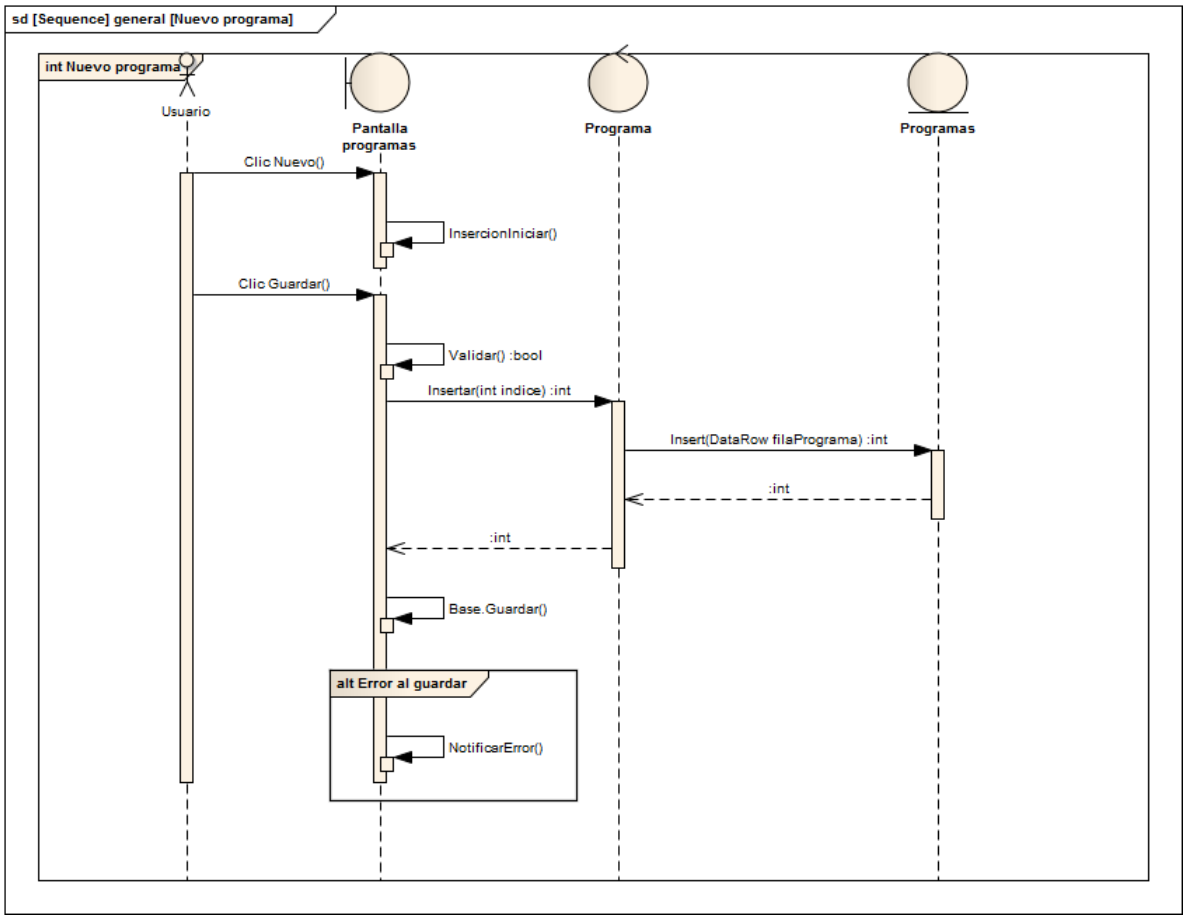
Tabla3.4: Caso de Uso Nuevo Programa.

UC Crear Nuevo Programa	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Programas** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.4: Diagrama de Secuencia Nuevo Programa



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.5. Editar Corte

El objetivo de este caso de uso es modificar un corte comercial existente.

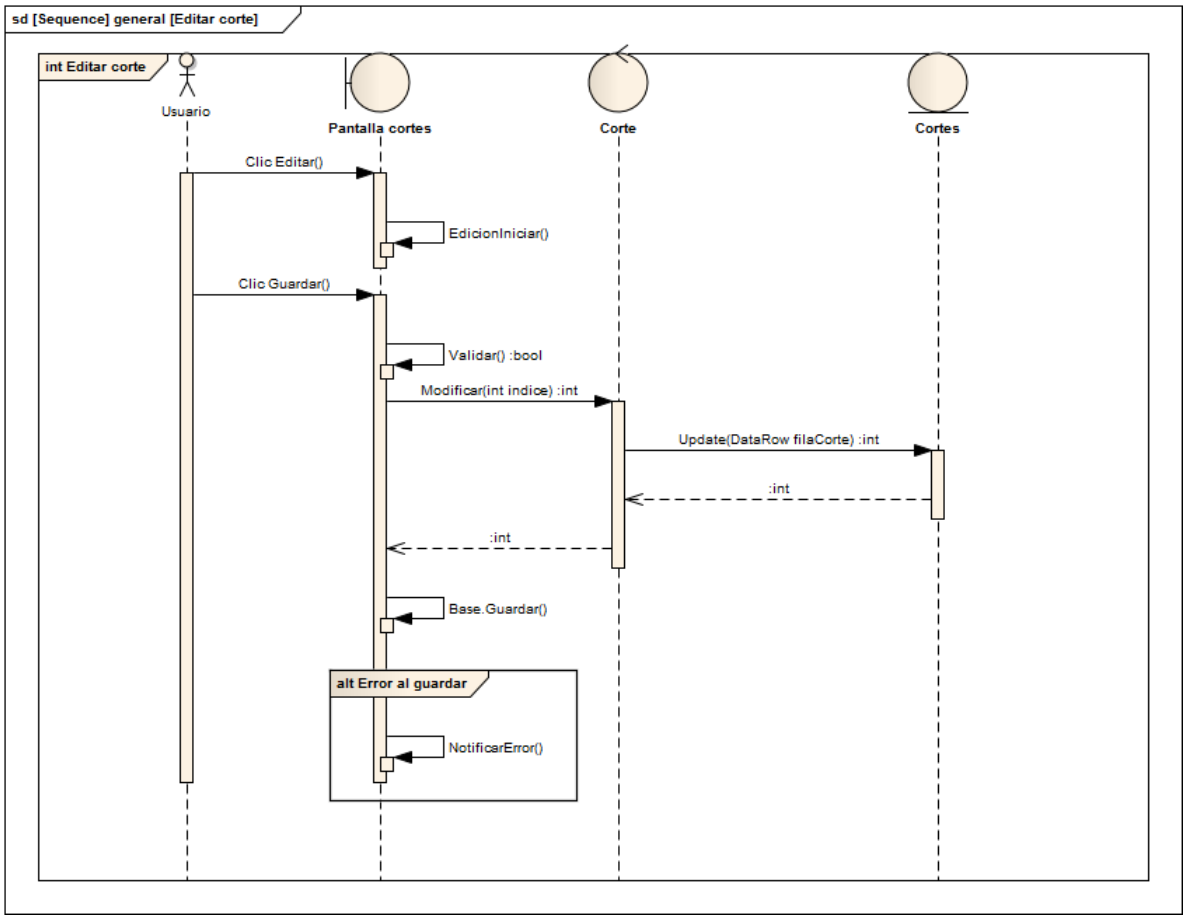
Tabla 3.5: Caso de Uso Editar Corte

UC Editar Corte	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Cortes** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.5: Diagrama de Secuencia Editar Corte



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.6. Editar Programa

El objetivo de este caso de uso es modificar unprograma existente.

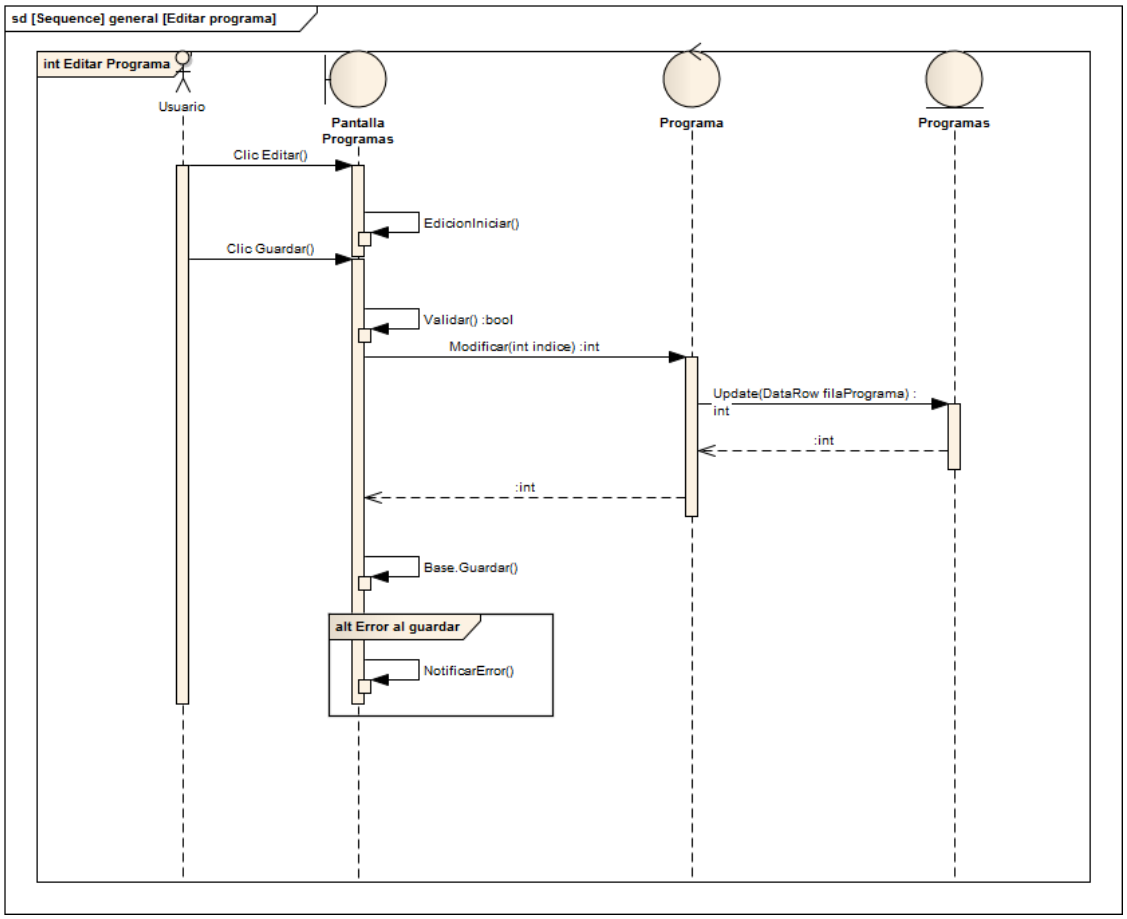
Tabla3.6: Caso de Uso Editar Programa

UC Editar Programa	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Programas** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.6: Diagrama de Secuencia Editar Programa



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.2.7. Eliminar Corte

El objetivo de este caso de uso es eliminar un corte comercial existente.

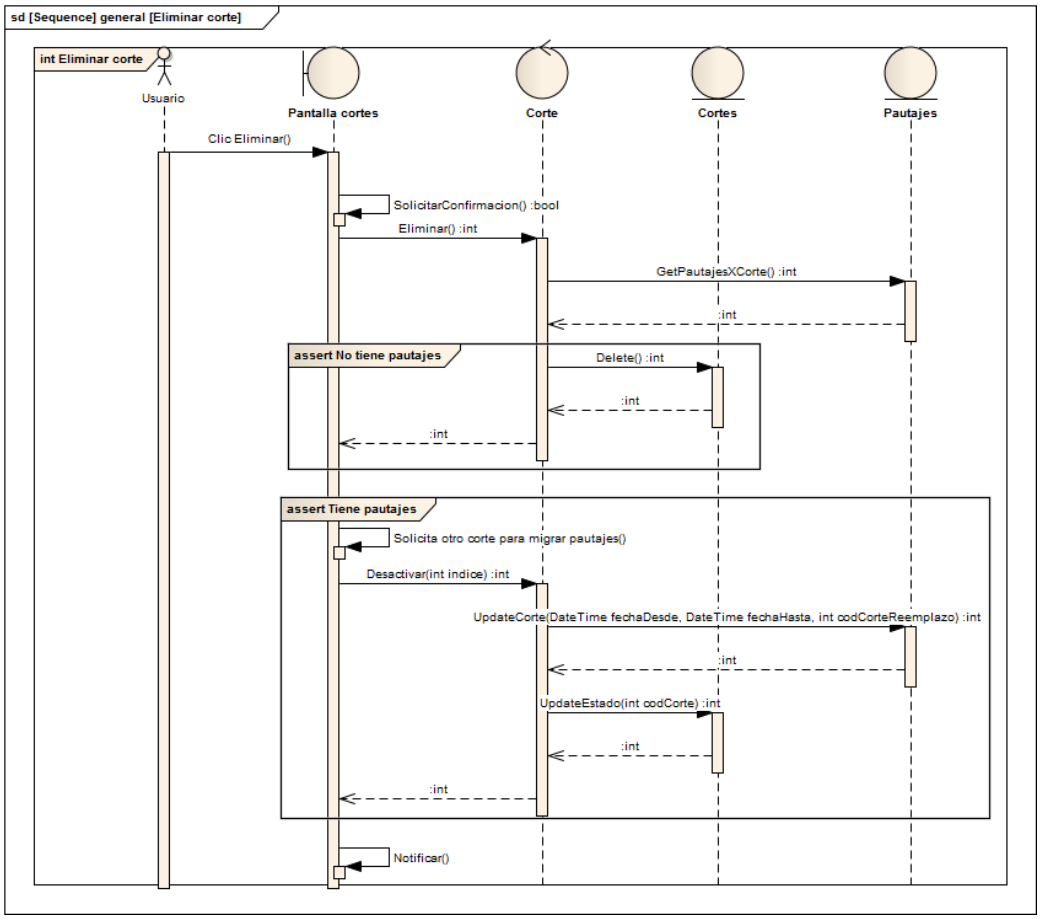
Tabla3.7: Caso de Uso Eliminar Corte

UC Eliminar Corte	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Cortes** y presiona el botón **Eliminar**. 2. El Sistema solicita confirmación. 3. El Sistema comprueba si el corte existe en la tabla **Pautajes** 4. El Sistema presenta mensaje.
•	Condición: El corte NO existe en la tabla Pautajes 1. El Sistema elimina el corte comercial de la base de datos. 2. Vuelve al paso 4 del Curso Normal.
•	Condición: El corte SI existe en la tabla Pautajes 1. El Sistema presenta una pantalla con una lista de cortes comerciales activos para migrar los pautajes relacionados con el corte que se va a eliminar. 2. El Usuario selecciona un corte comercial de la lista y un rango de fechas para efectuar la migración. 3. El Sistema cambia el estado de actividad del corte comercial 4. El Sistema reemplaza en la base de datos el código del corte comercial con el código del corte comercial escogido para la migración. 5. Vuelve al paso 4 del Curso Normal

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.7: Diagrama de Secuencia Eliminar Corte



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3. Casos de Uso y Diagramas de Secuencia de Gestión

3.1.3.1. Nuevo cliente

El objetivo de este caso de uso es crear un nuevo cliente.

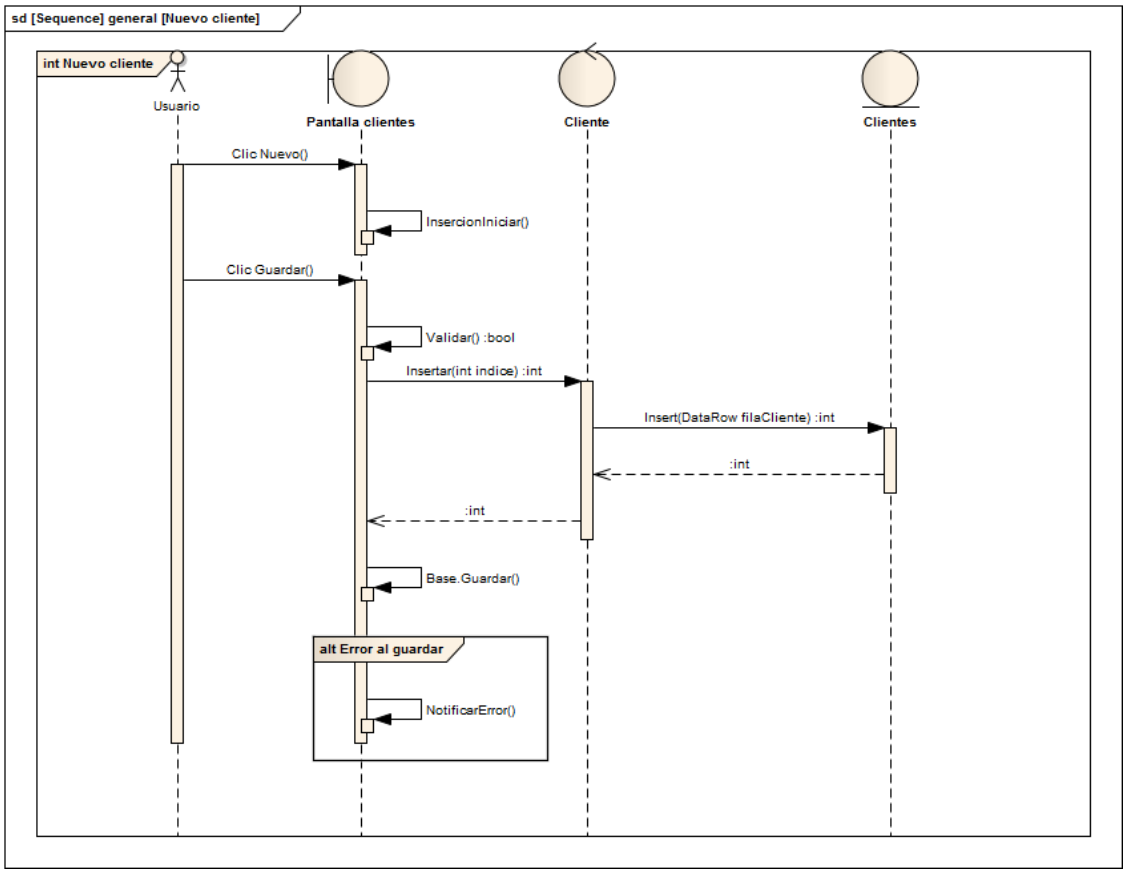
Tabla3.8: Caso de Uso Nuevo Cliente

UC Crear Nuevo Cliente	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Clientes** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 4. El Sistema captura una excepción. 5. El Sistema presenta mensaje de error. 6. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.8: Diagrama de Secuencia Nuevo Cliente



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.2. Nueva agencia

El objetivo de este caso de uso es crear una nueva agencia.

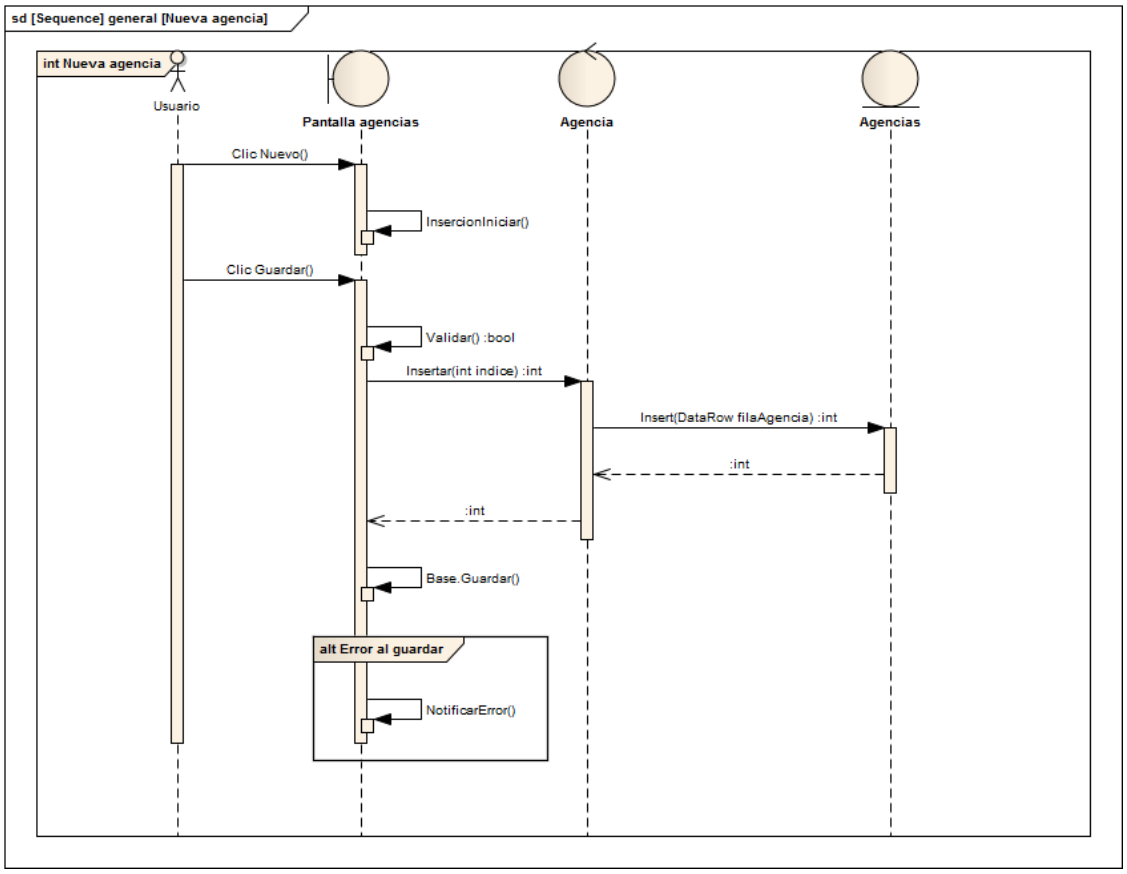
Tabla3.9: Caso de Uso Nueva Agencia

UC Crear NuevaAgencia	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Agencias** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.9: Diagrama de Secuencia Nueva Agencia



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.3. Nuevo tipo

El objetivo de este caso de uso es crear un nuevo tipo.

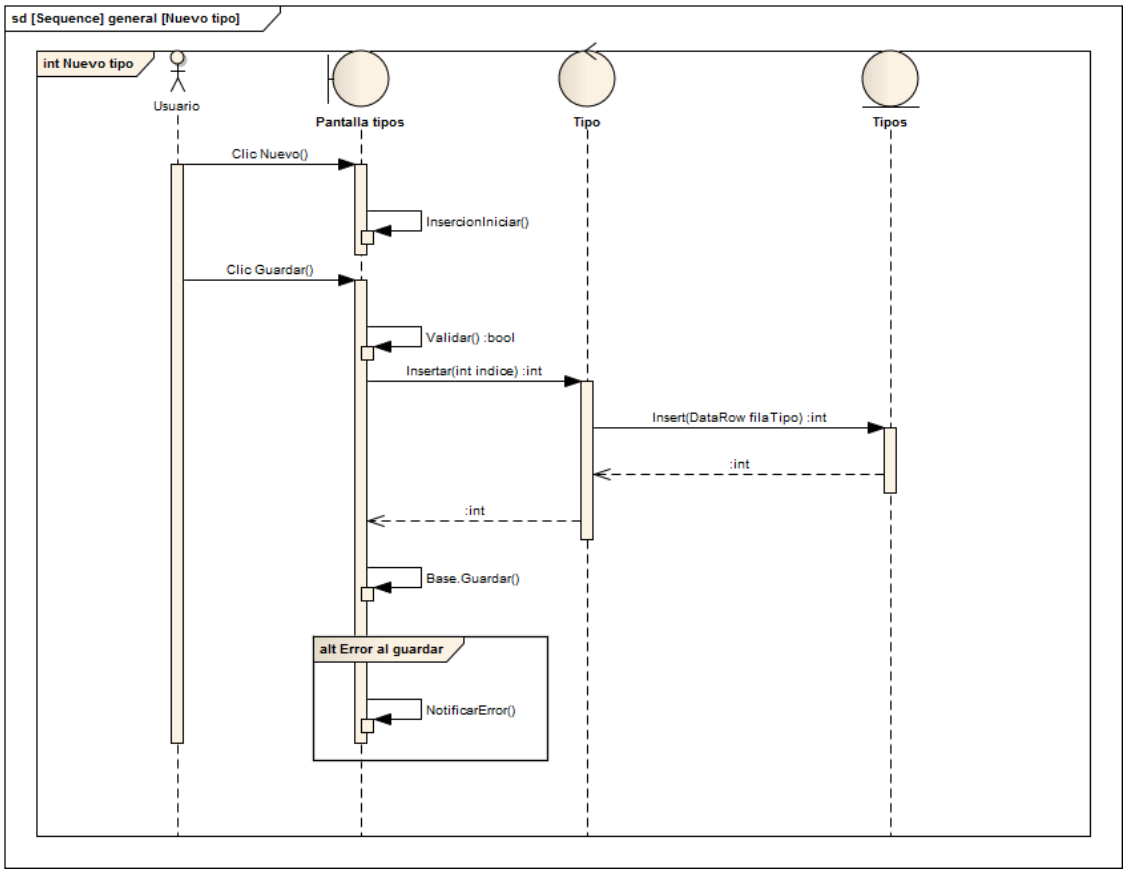
Tabla3.10: Caso de Uso Nuevo Tipo

UC Crear Nuevo Tipo	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Tipos** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.10: Diagrama de Secuencia Nuevo Tipo



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena**.**

3.1.3.4. Nueva categoría

El objetivo de este caso de uso es crear una nueva categoría.

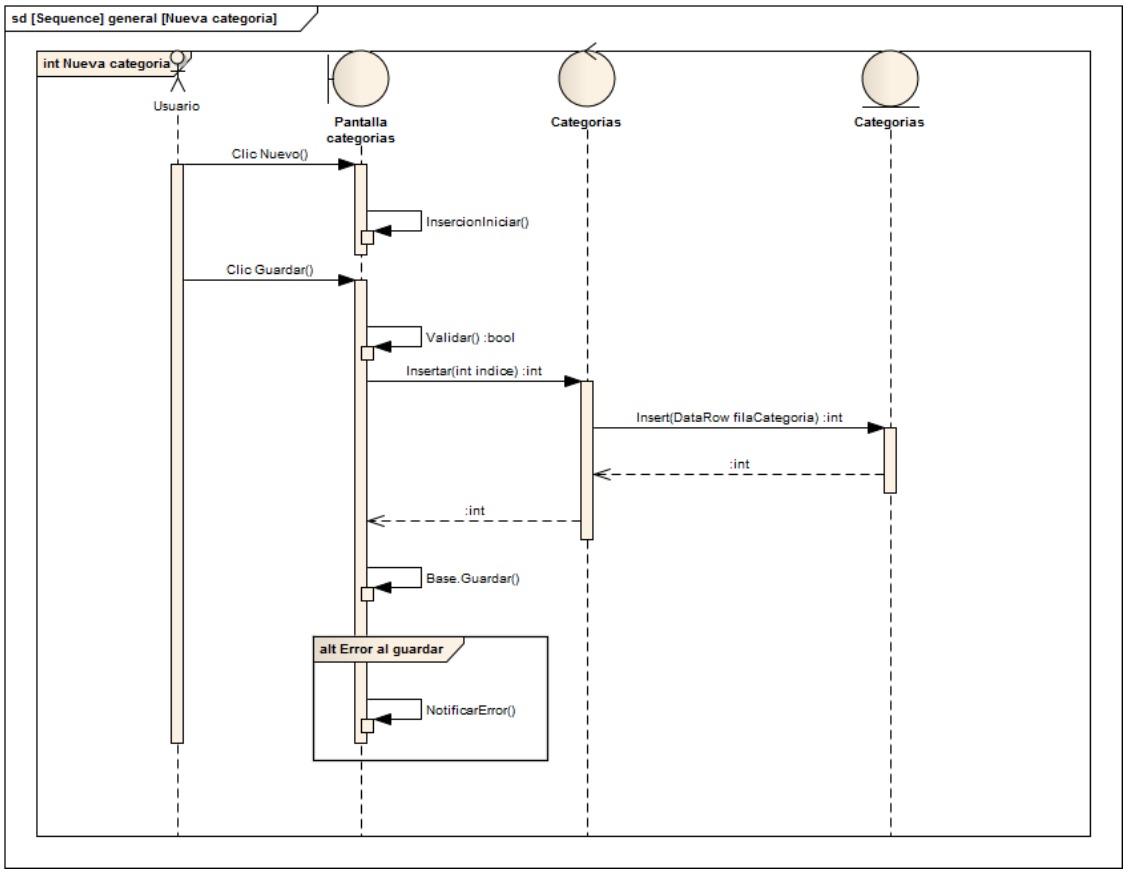
Tabla3.11: Caso de Uso Nueva Categoría

UC Crear Nueva Categoría	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Categorías** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.11: Diagrama de Secuencia Nueva Categoría



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.5. Editar cliente

El objetivo de este caso de uso es modificar un cliente existente.

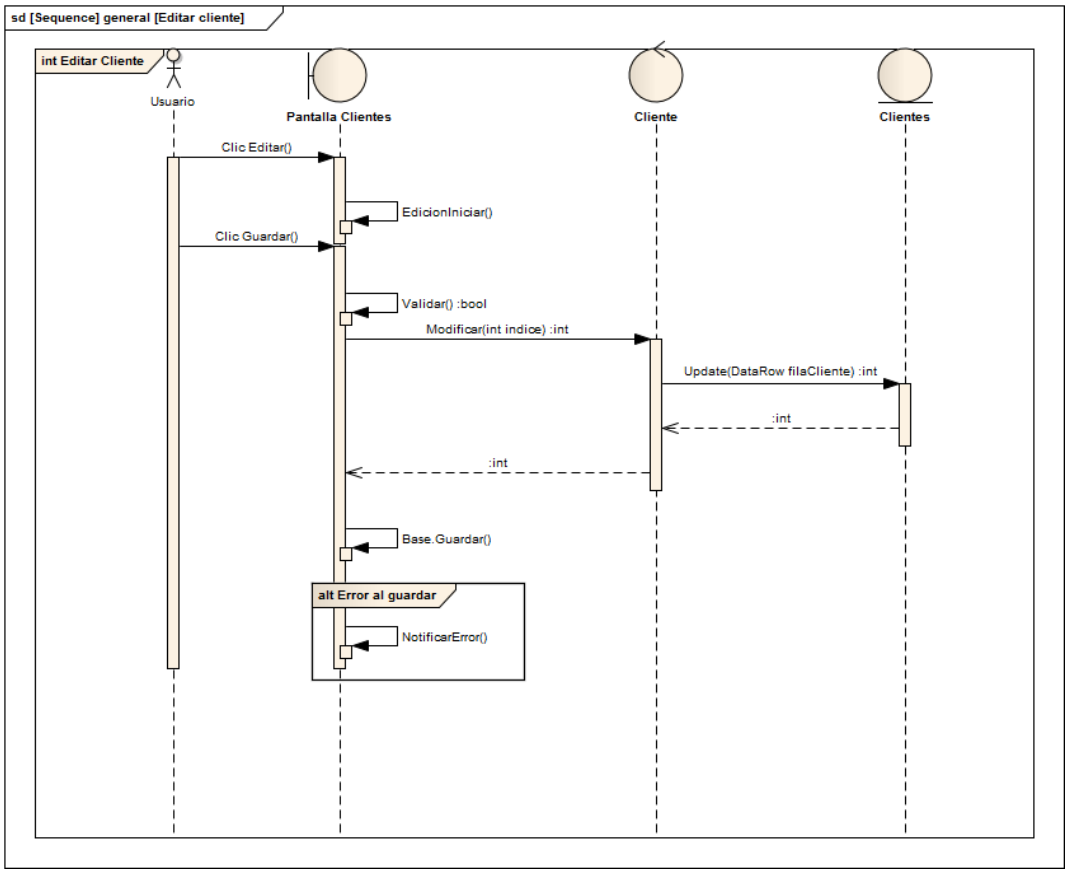
Tabla3.12: Caso de Uso Editar Cliente

UC Editar Cliente	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Clientes** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.12: Diagrama de Secuencia Editar Cliente



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.6. Editar agencia

El objetivo de este caso de uso es modificar una agencia existente.

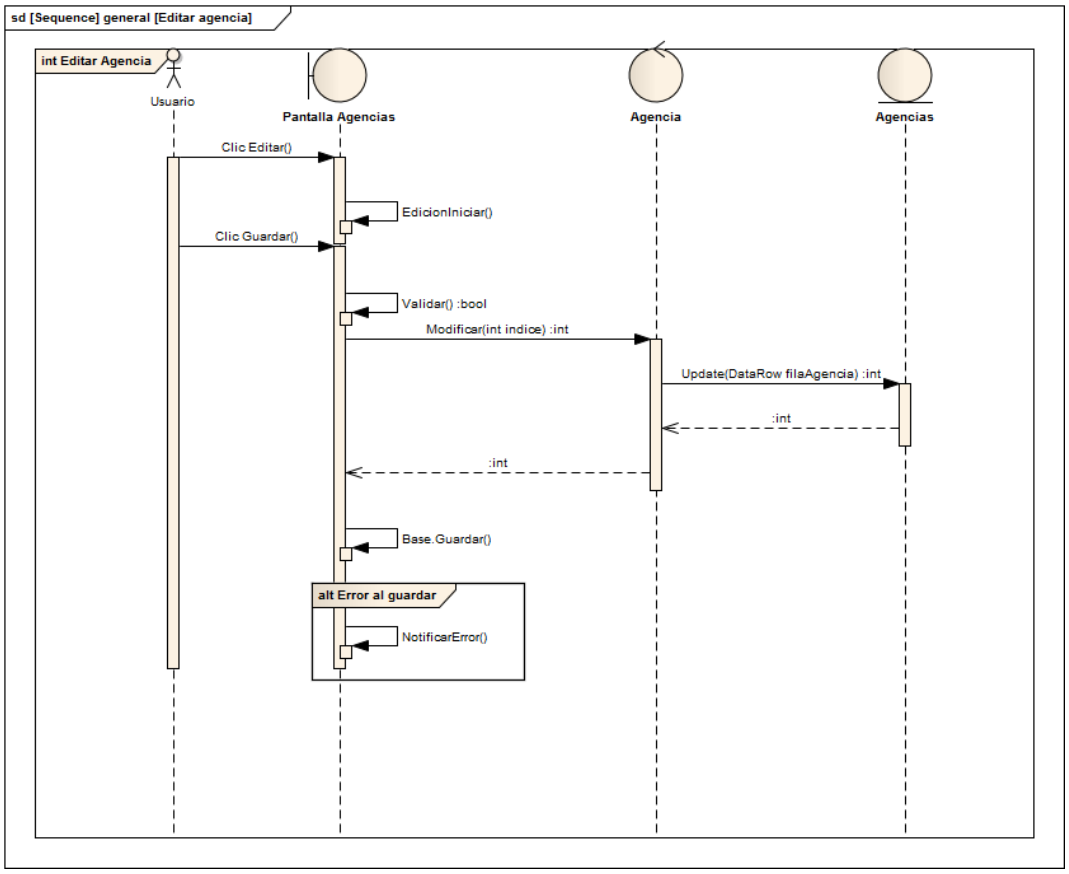
Tabla 3.13: Caso de Uso Editar Agencia

UC Editar Agencia	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Agencias** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.13: Diagrama de Secuencia Editar Agencia



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.7. Editar tipo

El objetivo de este caso de uso es modificar un tipo existente.

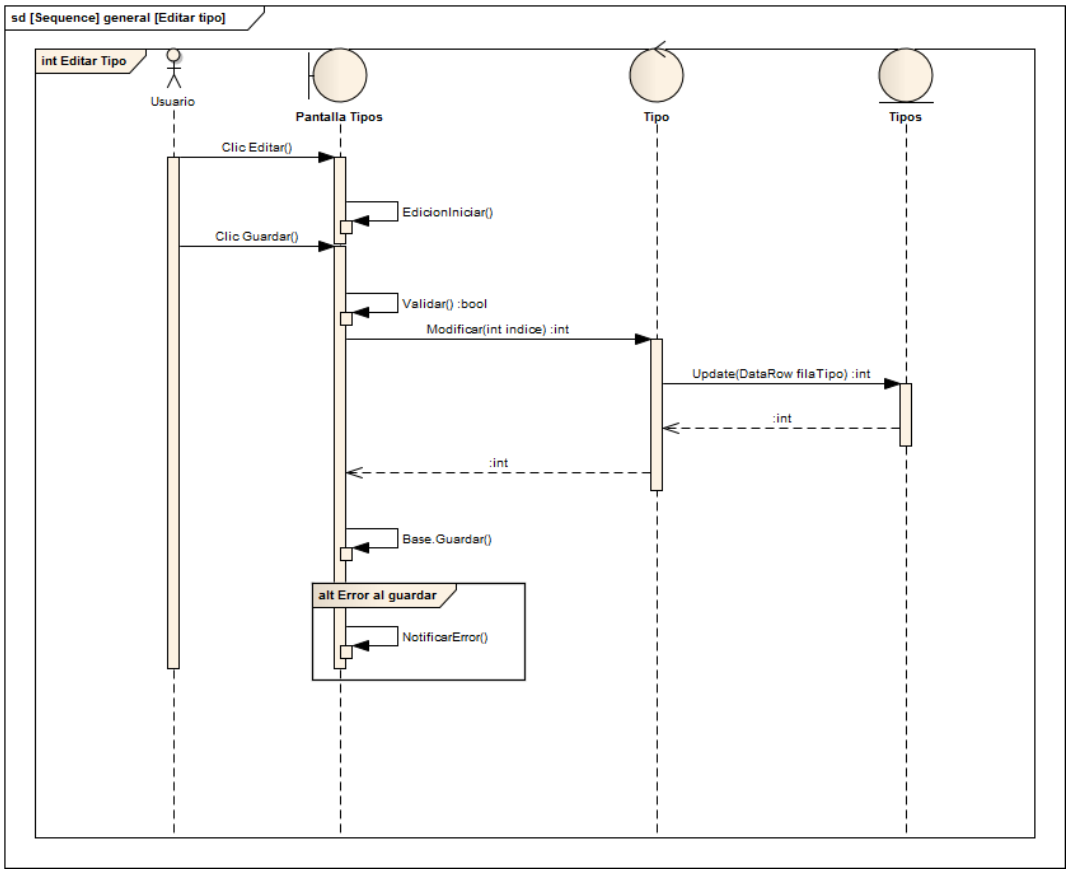
Tabla 3.14: Caso de Uso Editar Tipo

UC Editar Tipo	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Tipos** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena**.**

Figura 3.14: Diagrama de Secuencia Editar Tipo



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.3.8. Editar categoría

El objetivo de este caso de uso es modificar una categoría existente.

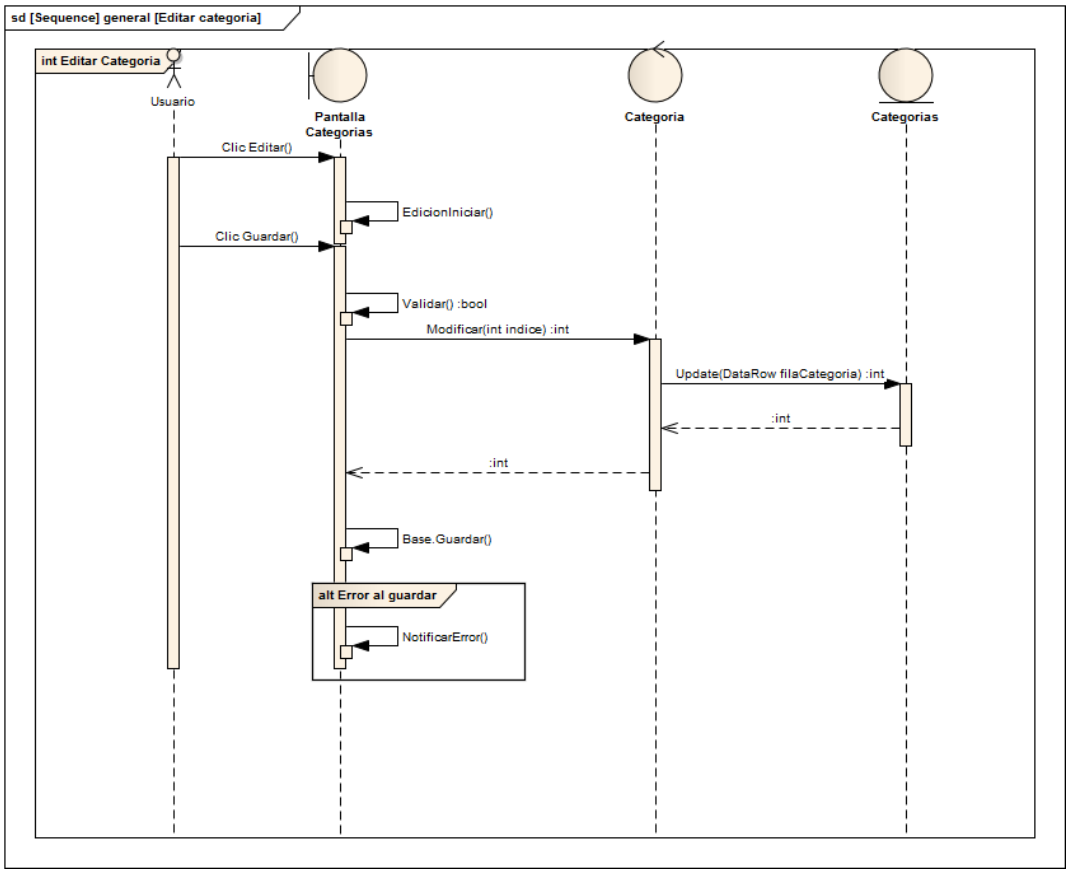
Tabla 3.15: Caso de Uso Editar Categoría

UC Editar Categoría	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Categorías** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.15: Diagrama de Secuencia Editar Categoría



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.4. Casos de Uso y Diagramas de Secuencia de Publicidad

3.1.4.1. Nuevo spot

El objetivo de este caso de uso es crear un nuevo spot.

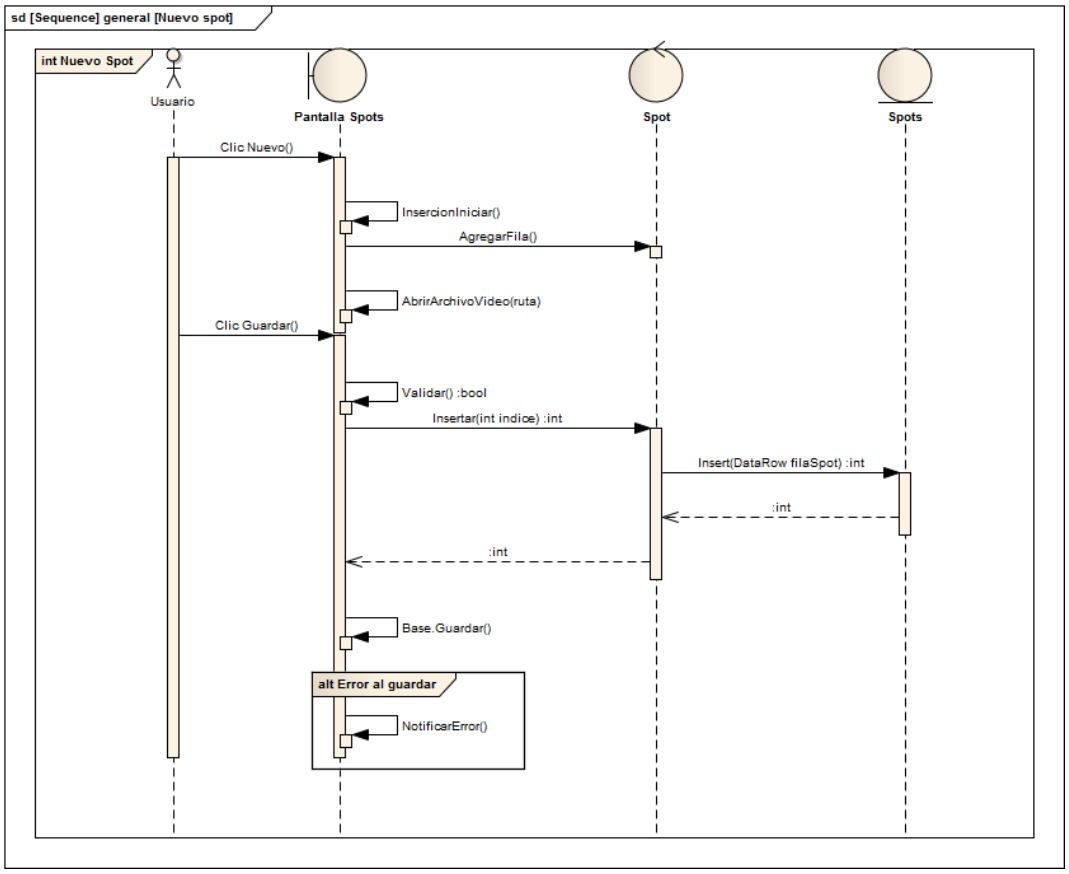
Tabla 3.16: Caso de Uso Nuevo Spot

UC Crear Nuevo Spot	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Nuevo**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario selecciona un archivo de video. 5. El Usuario presiona el botón **Guardar**. 6. El Sistema valida los datos. 7. El Sistema guarda en la base de datos. 8. El Sistema presenta mensaje. 9. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 5 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.16: Diagrama de Secuencia Nuevo Spot



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.4.2. Nueva Orden de Emisión

El objetivo de este caso de uso es crear una nueva orden de emisión para un spot.

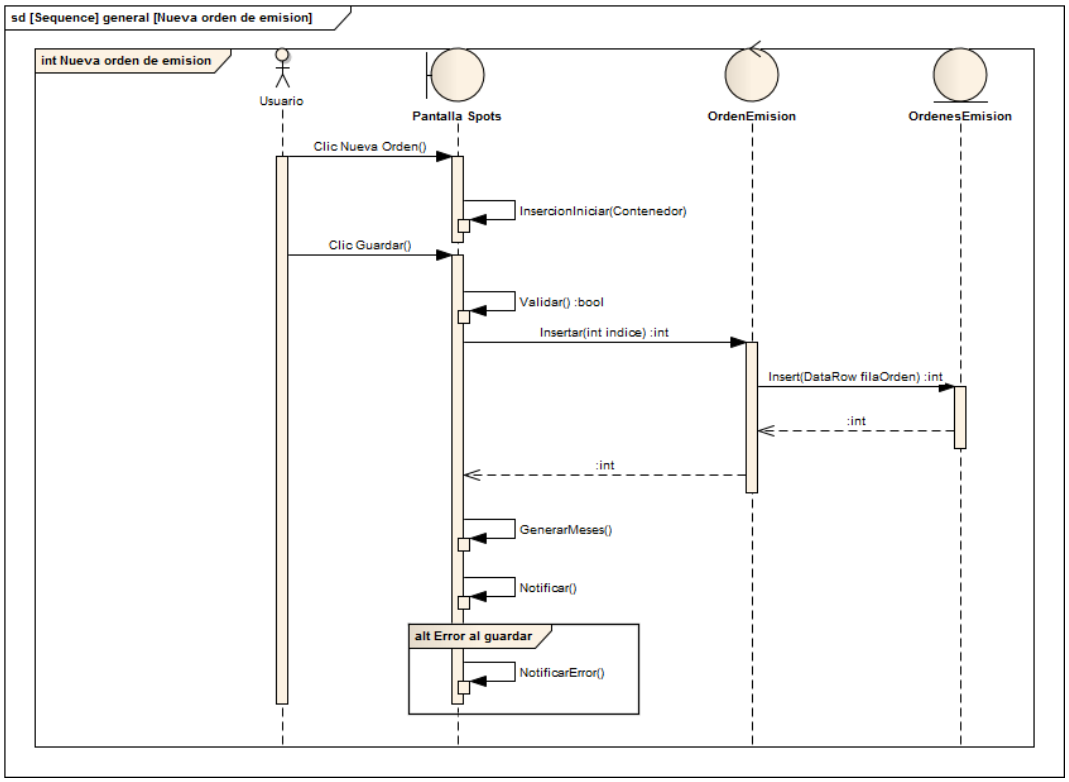
Tabla 3.17: Caso de Uso Nueva Orden de Emisión

UC Nueva Orden de Emisión	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Nueva Orden**. 2. El Sistema habilita los campos necesarios. 3. El Usuario ingresa los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.17: Diagrama de Secuencia Nueva Orden de Emisión



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.4.3. Editar spot

El objetivo de este caso de uso es modificar un spot existente.

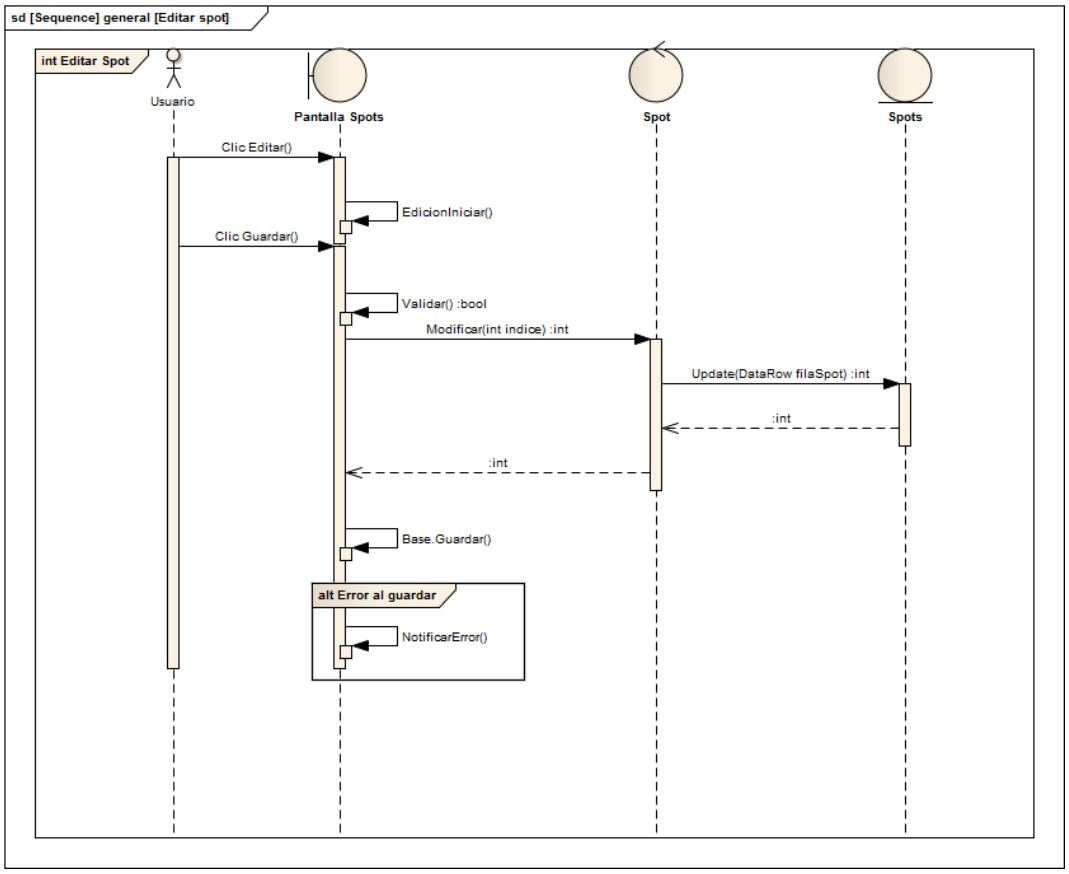
Tabla 3.18: Caso de Uso Editar Spot

UC Editar Spot	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Editar**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.18: Diagrama de Secuencia Editar Spot



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena**.**

3.1.4.4. Editar orden de emisión

El objetivo de este caso de uso es modificar una orden de emisión existente.

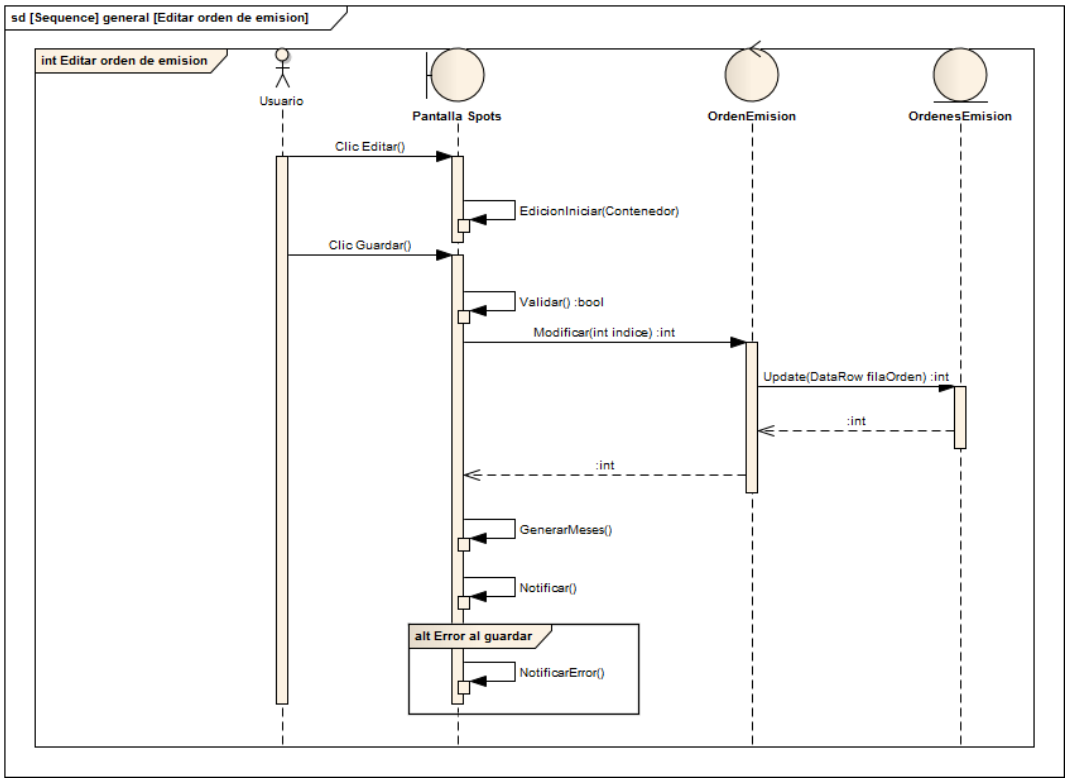
Tabla 3.19: Caso de Uso Editar Orden de Emisión

UC Editar Orden de Emisión	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Editar Orden**. 2. El Sistema habilita los campos necesarios. 3. El Usuario modifica los datos. 4. El Usuario presiona el botón **Guardar**. 5. El Sistema valida los datos. 6. El Sistema guarda en la base de datos. 7. El Sistema presenta mensaje. 8. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 1. El Sistema captura una excepción. 2. El Sistema presenta mensaje de error. 3. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena**.**

Figura 3.19: Diagrama de Secuencia Editar Orden de Emisión



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.4.5. Eliminar spot

El objetivo de este caso de uso es eliminar un spot existente.

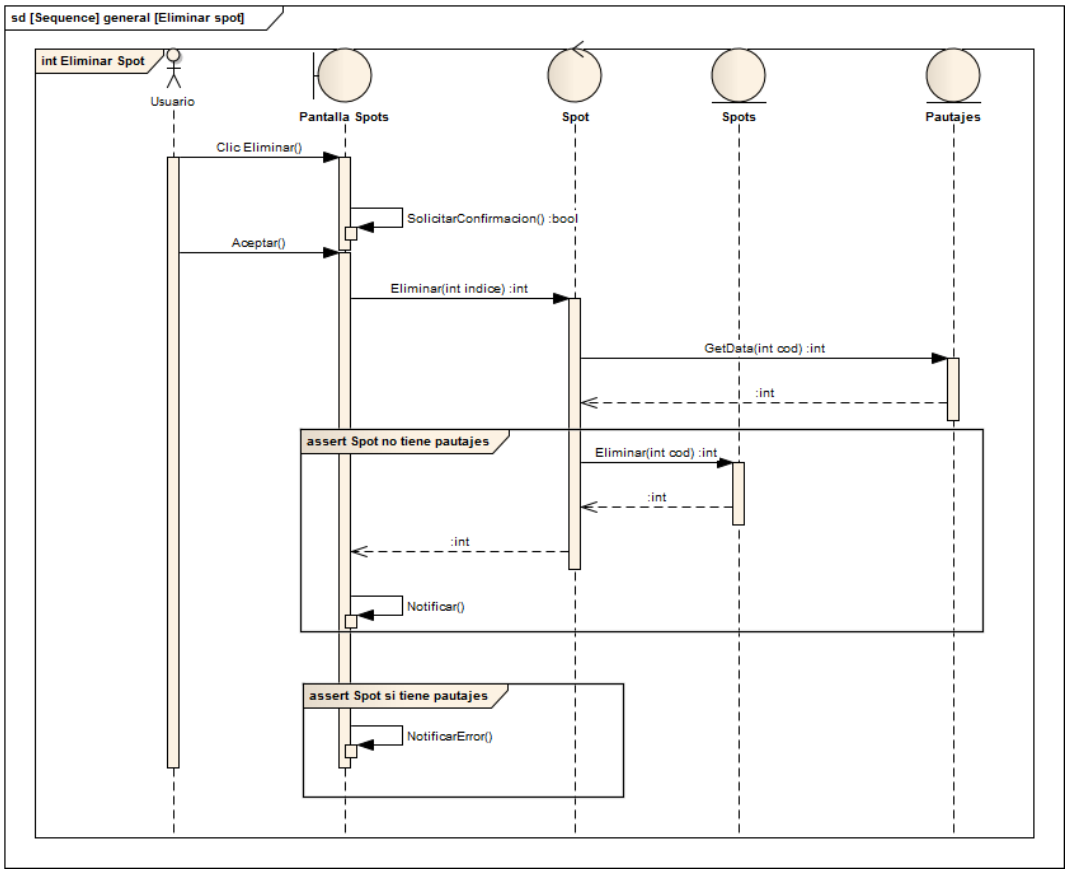
Tabla3.20: Caso de Uso Eliminar Spot

UC Eliminar Spot	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Eliminar**. 2. El Sistema solicita confirmación. 3. El Sistema verifica si el spot se encuentra en la tabla **Pautajes**. 4. El Sistema presenta mensaje.
•	Condición: El spot NO se encuentra en la tabla Pautajes 1. El Sistema elimina el spot de la base de datos 2. Vuelve al paso 4 del Curso Normal.
•	Condición: El spot SI se encuentra en la tabla Pautajes 1. El Sistema cambia el estado de actividad del spot. 2. Vuelve al paso 4 del Curso Normal.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.20: Diagrama de Secuencia Eliminar Spot



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.4.6. Emitir publicidad

El objetivo de este caso de uso es emitir la publicidad programada o agendada.

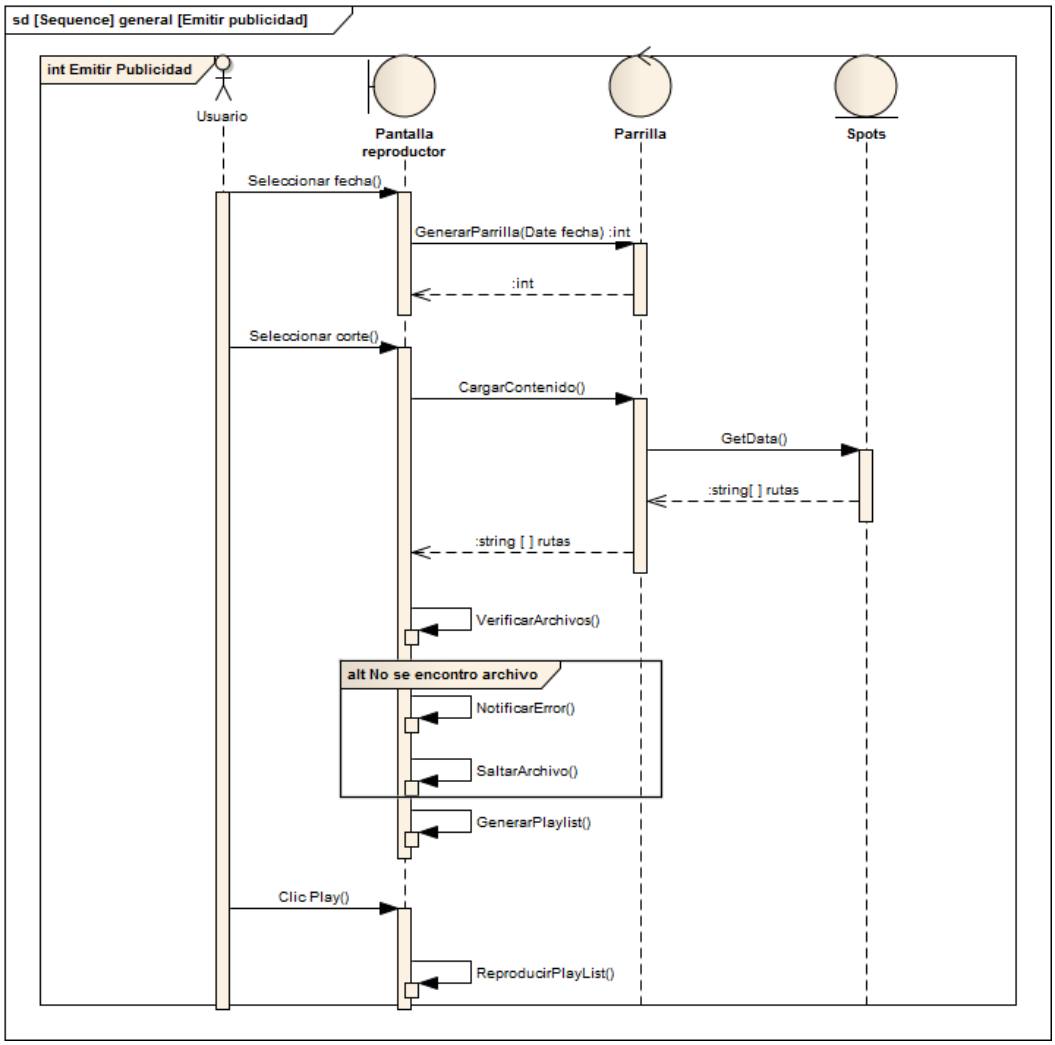
Tabla3.21: Caso de Uso Emitir Publicidad

UC Emitir Publicidad	
•	Curso Normal 1. El Usuario se encuentra en la **pantalla Reproductor** y selecciona una fecha del calendario. 2. El Sistema genera la parrilla para la fecha escogida. 3. El Usuario selecciona un corte de la lista generada. 4. El Sistema carga el contenido del corte y obtiene una lista. 5. El Sistema verifica las rutas de los spots de la lista y genera un PlayList de WMP. 6. El Usuario presiona el botón **Reproducir**. 7. El Sistema reproduce el PlayList
•	Alternativo: No se encontró archivo 1. El Sistema presenta mensaje de error. 2. El Sistema continúa con el siguiente archivo de la lista.

Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Figura 3.21: Diagrama de Secuencia Emitir Publicidad

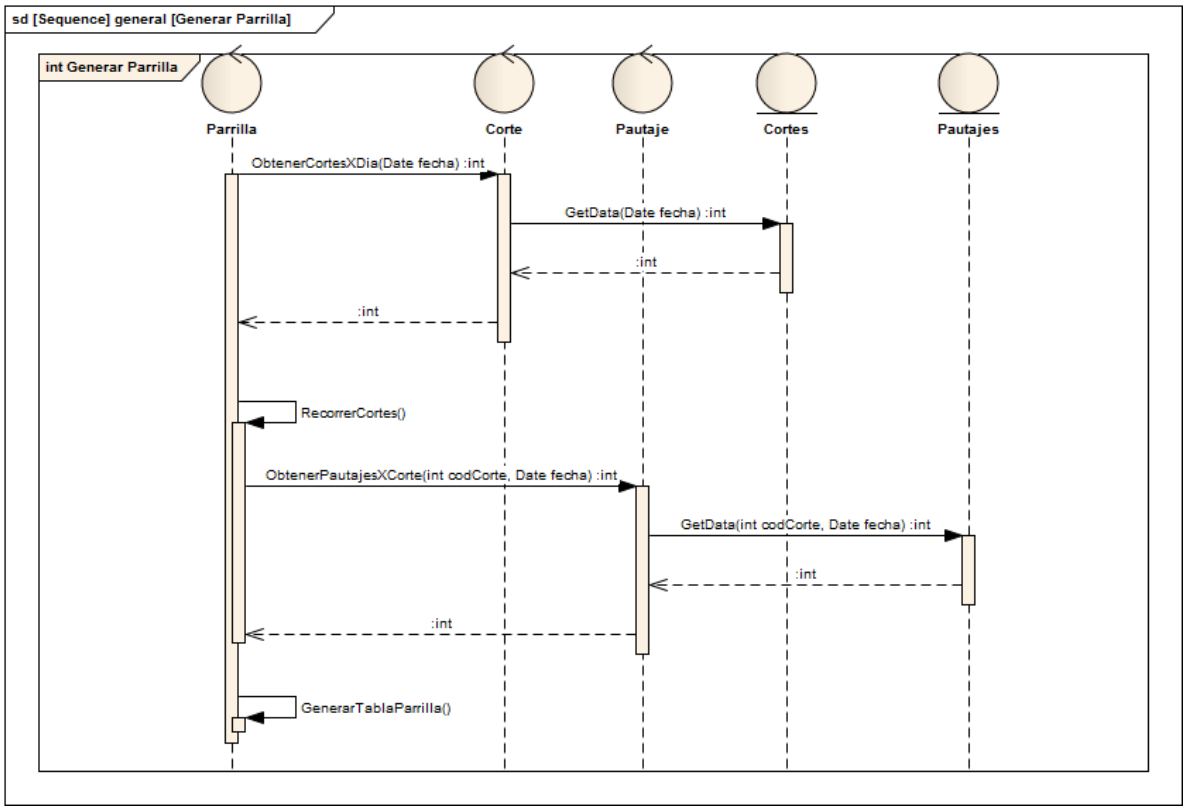


Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

Se ilustra a continuación el procedimiento de Generar Parrilla asociado al caso de uso abstracto con el mismo nombre.

Figura 3.22: Diagrama de Secuencia Generar Parrilla

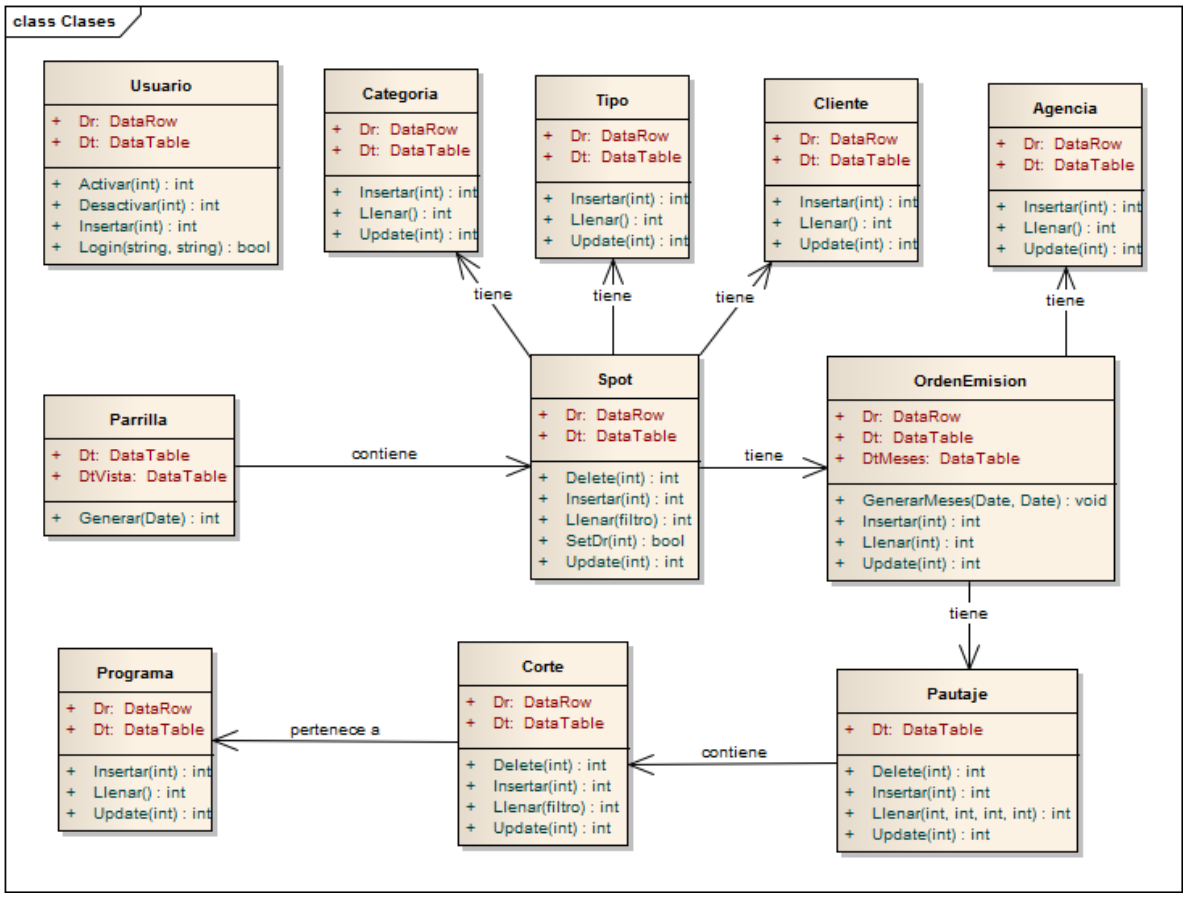


Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

3.1.5. Diagrama de Clases

Figura 3.23: Diagrama de clases



Fuente: Personal UV TELEVISIÓN.

Autor: Alan Yamil Correa Requena.

4. Implementación

4.1. Definición de la Arquitectura

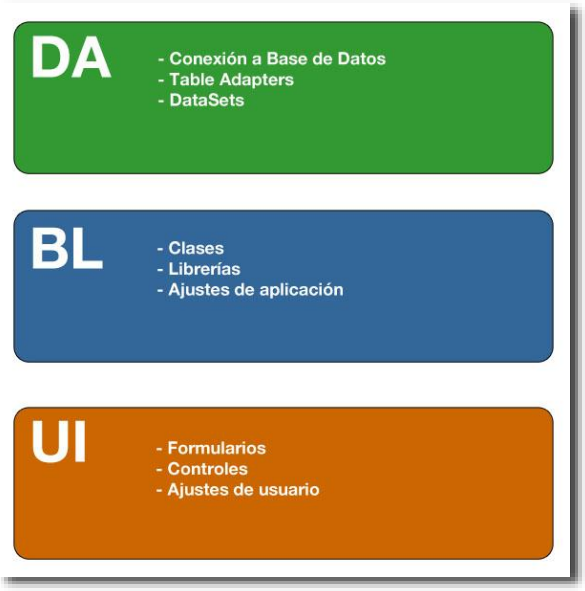
La Arquitectura de Software, a semejanza de los planos de la arquitectura de construcción, incluye guías generales que indican la estructura, funcionamiento e interacción entre las partes del software. La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, el conjunto de patrones y abstracciones coherentes que proporcionan el marco de trabajo se deben definir en base a objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. (González, 2014)

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Para el presente proyecto se utilizó la Arquitectura de Tres Niveles, donde cada nivel o capa tiene un reparto claro de funciones.

4.1.1. Capas.

Cada capa tiene elementos propios y se definen de la siguiente manera.

Figura 4.1: Esquema de capas



4.1.1.1. UI (User Interface)

Es la capa de Interfaz de Usuario donde se gestiona la interacción entre el Sistema y el Usuario. En las pantallas de interactividad se presentan los datos y se receptan las acciones de los usuarios. En esta capa se construyen los formularios (Windows Forms) y controles de usuario que constituirán las pantallas de interacción.

4.1.1.2. BL (Business Logic)

En este nivel se maneja la lógica de negocio. Es referenciada por la UI para sustentar las acciones generadas por esa capa y como fuente de datos para presentación y reflejo de cambios. En la BL se construyen las clases que contienen métodos con funcionalidad para realizar la gestión de los distintos módulos

4.1.1.3. DA (Data Access).

Componente de Acceso a Datos, sirve para interactuar con la Base de Datos a través de Table Adapters definidos en DataSets que se construyen en esta capa. Es referenciada por la BL para sustentar las operaciones que incluyan consultas y actualizaciones contra la Base de Datos.

4.2. Desarrollo del Software

El Desarrollo de Software es un proceso enfocado a brindar solución a un determinado problema mediante la construcción de un sistema o software.

Orientado por los objetivos planteados y definidos a lo largo de las fases establecidas por la metodología usada, el proceso de desarrollo de software es todo aquello que realizan los desarrolladores desde que se plantea el problema hasta que se implementa la solución. Comúnmente las metodologías incluyen las fases de Análisis, Diseño, Construcción e Implementación, repartidas o definidas de acuerdo a las particularidades y métodos de cada metodología específica.

4.2.1. Definición de Estándares de Programación.

Como en todo proceso, en el desarrollo del software se deben definir estándares de diseño y codificación para lograr un producto coherente y de mantenimiento sustentable.

Mediante el establecimiento de un estándar, se puede comprender a primera vista el trabajo de un desarrollador porque ha diferenciado la forma de representar diagramas, de declarar variables, de nombrar métodos, etc. Sobre todo cuando más de una persona está involucrada en el proceso de desarrollo, es importante definir a priori un conjunto de reglas y convenciones que se usarán en el diseño y codificación del proyecto.

4.2.1.1. Estándar para la Base de Datos

Los nombres de las tablas se escriben con mayúscula sin espacios, como separador se usará el guión bajo de ser necesario. Los nombres de los campos de las tablas, se escribirán en minúsculas usando abreviaturas definidas y el guión bajo como separador de palabras.

Ejemplo: spot_id

Las claves primarias son de tipo entero (int) y son auto numéricos, es decir se incrementan automáticamente con cada nuevo registro.

Los nombres de las claves primarias empezarán con el prefijo pk_, los nombres de las claves foráneas empezarán con el prefijo fk_ y los nombres de las claves únicas empezarán con el prefijo uk_.

Los campos de texto de longitud pequeña y moderada serán de tipo varchar, los de longitud grande serán de tipo text.

Tipos de datos.

Tabla4.1: Tipos de datos a usar en el proyecto

Descripción	**Tipo de dato**
Numérico	INT
Decimales	FLOAT
Caracteres	CHAR
Fecha hora	DATETIME
Cadena de texto	VARCHAR
Cadena de texto larga	TEXT

Fuente: Visual Estudio 2012
Autor: Alan Correa

4.2.1.2. Estándar para la codificación en C#

Para la estructura de las clases, de los tableAdapters, de los formularios y controles de usuario, así como la nomenclatura de todos los elementos usados en la codificación, se usan las siguientes convenciones y reglas.

Nombramiento Pascal: La primera letra de todas las palabras es mayúscula, el resto, minúscula.

Nombramiento Camel: La primera letra de todas las palabras excepto la primera palabra es mayúscula, el resto, minúscula.

Para los nombres de las clases, usamos nombramiento Pascal:

```
publicclassOrdenEmision
{
    ...
}
```

Para los nombres de los métodos usamos nombramiento Pascal:

```
voidLlenarPorFiltro(filtro filtro)
{
    ...
}
```

Para los nombres de variables y parámetros usamos nombramiento Camel:

```
intnumeroRegistros = 0;
intInsertarSpot(intindice)
{
    returnthis.ta.Insert(dt[indice].nombre, dt[índice].ruta…);
}
```

Los nombres de variables usadas como Flags, empiezan con un guion bajo y nombramiento Camel:

```
privatebool _auxNumRegistros;
```

Se deja una línea en blanco para separar grupos lógicos de código:

```
privatebool Validar()
{
        bool vale = false;

        this.errorProvider.Clear();

        if(spot.Dt[bsSpot.position].duracion == 0){errorProvider.SetError(…);}
        elseerrorProvider.Clear();

        return vale;
}
```

Los controles llevarán un prefijo de acuerdo a su tipo y según la siguiente tabla:

Tabla 4.2: Prefijos para nomenclatura de controles de interfaz

CONTROL	PREFIJO
Label	lbl
TextBox	txt
DataGridView	dgv
Button	btn
ComboBox	cb
ListBox	lst
CheckBox	chk
RadioButton	rb
DateTimePicker	dtp
NumericUpDown	Nud
GroupBox	Gb
TabControl	Tc
TabPage	Tp
DataSet	DS
DataSet como objeto de transporte	DTO
DataTable	Dt
DataRow	Dr

Fuente: Visual Estudio 2012

Autor: Alan Correa

Ejemplos: lblNombreSpot, txtRuta, btnGuardar, dgvPautajes

4.3. Conceptos Básicos

4.3.1. Microsoft Visual Studio 2012 Express

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. (visualstudio, 2014)

Visual Studio 2012 Express para Windows Desktop permite únicamente el desarrollo de aplicaciones en entorno Windows y no incluye las librerías para desarrollo Web. Esta es la versión de Visual Studio a utilizar en el presente proyecto.

4.3.2. Dataset

Los objetos DataSets, un grupo de clases que describen una simple base de datos relacional en memoria, las clases forman una jerarquía de contención:

Un objeto DataSet representa un esquema (o una base de datos entera o un subconjunto de una). Puede contener las tablas y las relaciones entre esas tablas.

Un objeto DataTable representa una sola tabla en la base de datos. Tiene un nombre, filas, y columnas.

Un objeto DataView "se sienta sobre" un DataTable y ordena los datos (como una cláusula "orderby" de SQL) y, si se activa un filtro, filtra los registros (como una cláusula "where" del SQL). Para facilitar estas operaciones se usa un índice en memoria. Todas las DataTables tienen un filtro por defecto, mientras que pueden ser definidos cualquier número de DataViews adicionales, reduciendo la interacción con la base de datos subyacente y mejorando así el desempeño.

Un DataColumn representa una columna de la tabla, incluyendo su nombre y tipo.

Un objeto DataRow representa una sola fila en la tabla, y permite leer y actualizar los valores en esa fila, así como la recuperación de cualquier fila que esté relacionada con ella a través de una relación de clave primaria - clave foránea.

Un DataRelation es una relación entre las tablas, tales como una relación de clave primaria - clave ajena. Esto es útil para permitir la funcionalidad del DataRow de recuperar filas relacionadas.

Un Constraint describe una propiedad de la base de datos que se debe cumplir, como por ejemplo que los valores en una columna de clave primaria deben ser únicos. A medida que los datos son modificados cualquier violación que se presente causará excepciones.

Los DataSet trabajan en un entorno “desconectado”, es decir, no necesitan de una conexión permanente o abierta con una base de datos, ya que los datos se cargan y mantienen en un espacio de memoria. (StevenPo, 2011)

Un DataSet es llenado desde una base de datos por un TableAdapter.

4.3.3. Tableadapter

Los TableAdapters comunican la aplicación con una base de datos. Más específicamente, un TableAdapter se conecta con una base de datos, ejecuta consultas o procedimientos almacenados, y devuelve una nueva tabla de datos rellena con los datos devueltos o rellena una DataTable existente con los datos devueltos. Los TableAdapters también se utilizan para devolver los datos actualizados desde la aplicación a la base de datos. (StevenPo, 2011)

Además de la funcionalidad estándar del control DataAdapter, los objetos TableAdapter proporcionan métodos con tipo adicionales que encapsulan consultas que comparten un esquema común con el control DataTable con tipo asociado. Dicho de otra forma, se puede tener tantas consultas como se desee en un TableAdapter, siempre y cuando devuelvan datos que cumplan el mismo esquema.

Los TableAdapters además se encargan de manejar las conexiones con las bases de datos, abriéndolas, cerrándolas y manejando el pool de conexiones de manera transparente para el programador.

4.3.4. SQL SERVER 2012 Express

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL. (Microsoft, s.f.)

SQL Server 2012 permite a los clientes crear aplicaciones críticas y soluciones Big Data mediante tecnología en memoria y de alto rendimiento a través de OLTP, almacenamiento de datos, Business Intelligence y cargas de trabajo analíticas sin tener que comprar costosos complementos ni aplicaciones de alta gama. SQL Server 2012 utiliza un conjunto de herramientas comunes para implementar y administrar bases de datos tanto en la nube como en el entorno local, lo que facilita que los clientes puedan aprovechar la nube con los conocimientos existentes.

- Soporte de transacciones.
- Soporta procedimientos almacenados.
- Incluye también un entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005, 2008 y 2012 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita.

Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa la base de datos (Microsoft SQL Server), con el entorno de desarrollo (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.

En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD, osql, o PowerShell.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos.

4.3.5. Software Ideas Modeller

Software Ideas Modeler es una herramienta CASE y un modelador UML. El modelador soporta los 14 tipos de diagramas especificados en UML 2.4. Entre otros, soporta los siguientesestándares y diagramas:

- DiagramasERD
- BPMN 2.0

- SysML
- ArchiMate
- JSD
- CRC
- Diagramas de flujo
- Mapasmentales

Software Ideas Modeler es obra del desarrollador DušanRodina. El software está escrito en C#. (Wikipedia, 2015)

Se distribuye en 3 versiones: Standard, Professional y Ultimate. El presente proyecto se desarrolló utilizando la versión profesional en modo evaluación.

4.3.6. CrystalReports

CrystalReports es una aplicación de inteligencia empresarial utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos (bases de datos).

Varias aplicaciones, como Microsoft Visual Studio, incluyen una versión OEM de CrystalReports como una herramienta de propósito general del informes/reportes.

CrystalReports se convirtió en el escritor de informes estándar cuando Microsoft lo liberó con Visual Basic.

4.4. Diseño

4.4.1. Diseño de base de Datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. Un modelo de datos es por tanto una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. Conceptualmente, una aplicación puede ser caracterizada por:

- Propiedades estáticas: entidades (u objetos), propiedades (o atributos) de esas entidades, y relaciones entre esas entidades.
- Propiedades dinámicas: operaciones sobre entidades, sobre propiedades o relaciones entre operaciones.
- Reglas de integridad sobre las entidades y las operaciones (claves primarias, únicas y foráneas).

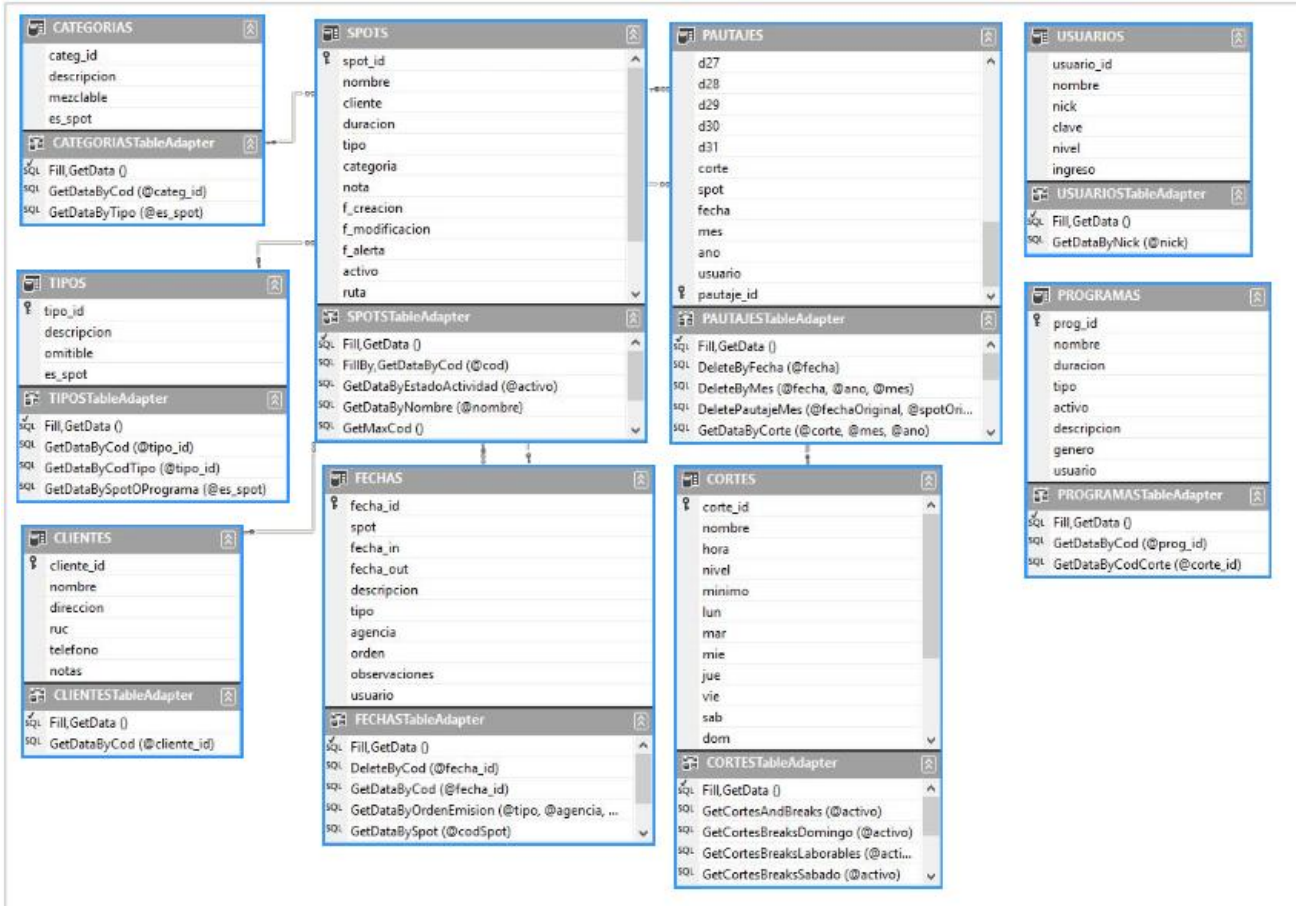
El diseño de la base de datos ha tomado como base el modelo de dominio para definir los tableSpaces y las tablas.

4.4.1.1. Definición de Tablas.

Las tablas en las bases de datos RDBMS se componen de columnas y filas. Las columnas representan una variable o campo, y las filas son tuplas, que relacionan datos de manera unívoca en un “registro”.

Las tablas definidas para este proyecto se organizan de la siguiente manera:

Figura 4.2: Diagrama de base de datos con TableAdapters



Fuente: Base de Datos Del Sistema.

Autor: Alan Yamil Correa Requena.

4.4.2. Construcción de la base de Datos

Para crear la base de datos se siguen los siguientes pasos:

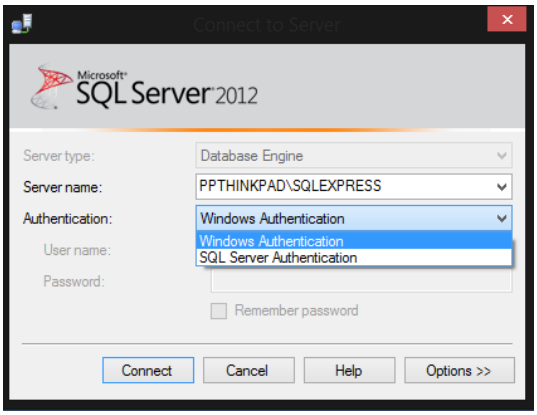
Se utilizara el gestor de base de datos SQL Server Management Studio 2012. Este gestor permite manejar tanto las versiones de pago como las versiones express de SQL Server.

Figura 4.3: Pantalla de inicio de SQL Server 2012 Management Studio

Fuente: Base de Datos Del Sistema.
Autor: Alan Yamil Correa Requena.

El primer paso es conectarse al servidor de BDD. En este caso se ha configurado el servicio SQLEXPRESS para conectarse usando modo mixto, es decir autenticación basada en las credenciales de Windows y la autenticación basada en usuarios de la base de datos. De modo que se conectara usando las credenciales de la cuenta de usuario de Windows.

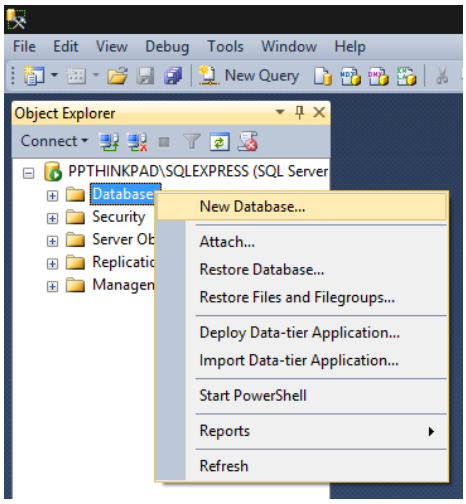
Figura 4.4: Diálogo de autenticación de SQL Server



Fuente: Base de Datos Del Sistema.
Autor: Alan Yamil Correa Requena.

Una vez que se conecte al servicio EQUIPO\SQLEXPRESS podemos crear una nueva base de datos o manejar una existente. Para crear una nueva BDD vamos al nodo Databases y del menú contextual escogemos New Database.

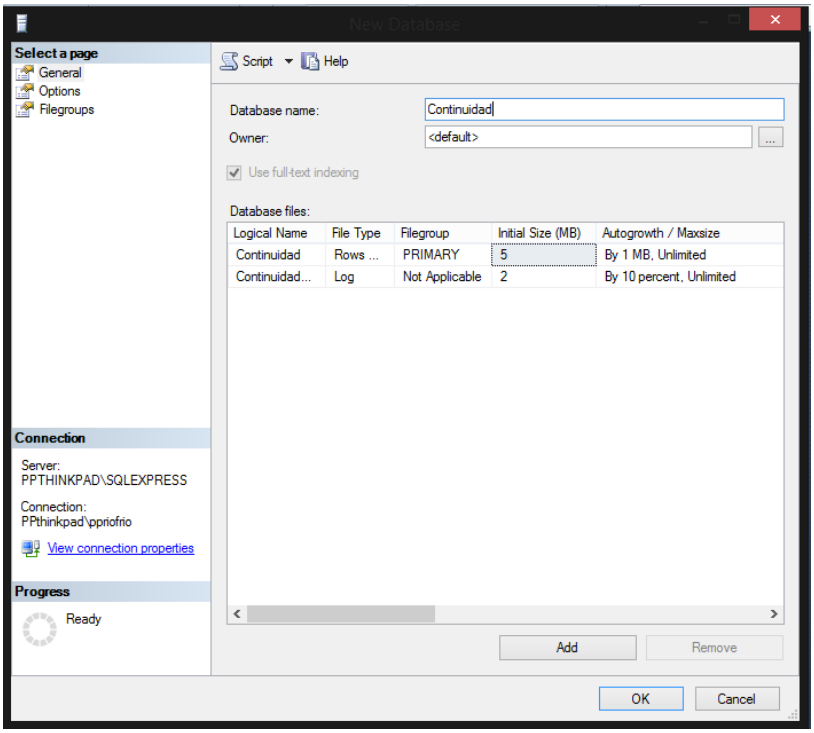
Figura 4.5: Creación de una nueva base de datos



Fuente: Base de Datos Del Sistema.
Autor: Alan Yamil Correa Requena.

Solamente necesitamos especificar el nombre que va a tener la base de datos, el resto de parámetros son opcionales y nos permiten escoger donde se guardará el datafile (.mdf) y el tamaño de archivo inicial.

Figura 4.6: Parámetros de creación de la base de datos

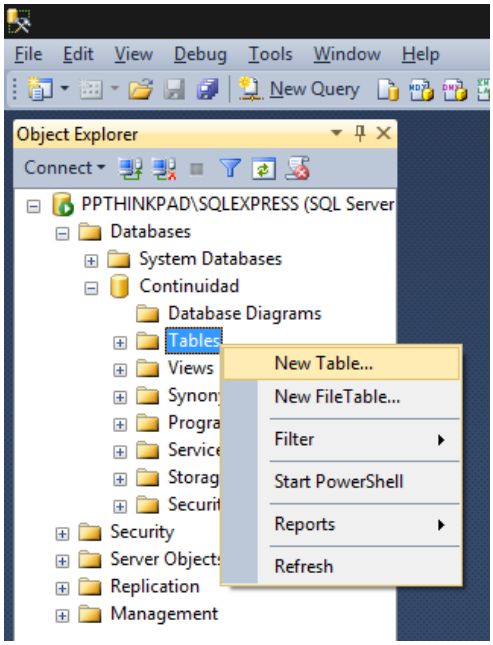


Fuente: Base de Datos Del Sistema.

Autor: Alan Yamil Correa Requena.

Para crear las tablas de la base de datos, damos clic derecho sobre el nodo Tables y seleccionamos New Table.

Figura 4.7: Creación de una nueva tabla

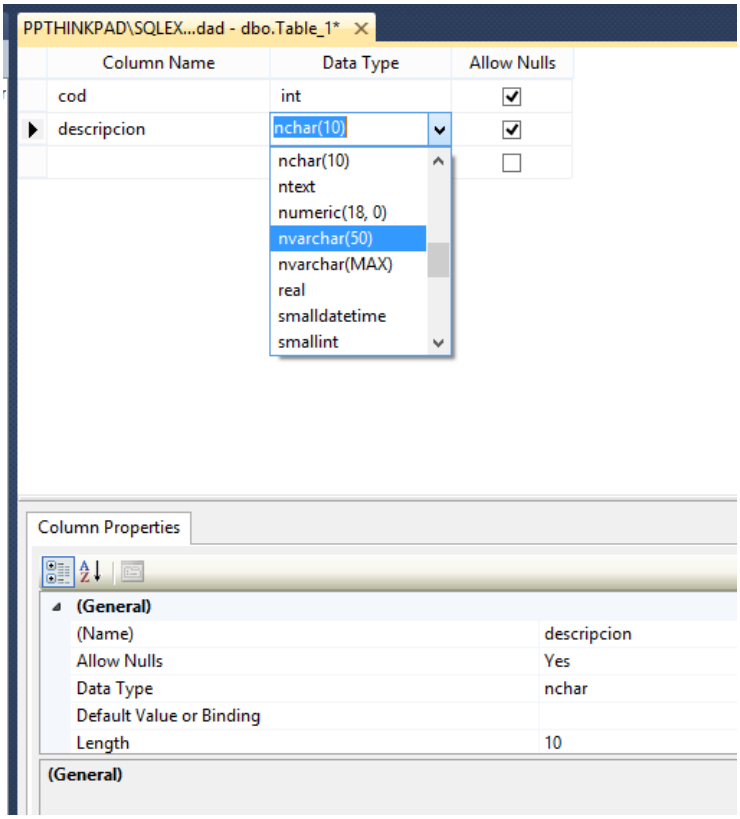


Fuente: Base de Datos Del Sistema.

Autor: Alan Yamil Correa Requena.

Aparece un panel donde iremos colocando los nombres y tipos de los campos que contendrá la tabla. Al finalizar se presiona el botón Guardar o Save y en ese momento daremos un nombre a la tabla. Nótese que en la parte inferior tenemos un sub panel que nos permite definir los valores por defecto, las claves primarias, si permite tener valores nulos, si será autoincrementable, etc.

Figura 4.8: Creación de campos en una tabla



Fuente: Base de Datos Del Sistema.

Autor: Alan Yamil Correa Requena.

Es recomendable incluir siempre claves primarias y claves foráneas en las tablas porque éstas nos permitirán mantener la integridad referencial de los datos.

Los triggers o disparadores son muy útiles al momento de programar bases de datos y también ayudan a mantener la integridad referencial. Sin embargo, para el presente proyecto manejaremos la integridad referencial desde el código y no directamente en la BDD.

4.4.3. Construcción del Proyecto En Visual Studio 2012 Express

Visual Studio organiza los proyectos en contenedores conocidos como soluciones. Las soluciones contienen uno o varios proyectos, los proyectos contienen elementos que pueden estar organizados en carpetas. La aplicación de este proyecto se realizará bajo una solución con 3 proyectos correspondientes a las capas UI, BL y DAC.

4.4.3.1. Creación y organización de la solución.

Para crear la solución y sus proyectos, se siguen los siguientes pasos:

Se abre Microsoft Visual Studio 2012, en este caso, la versión Express para aplicaciones de escritorio de Windows.

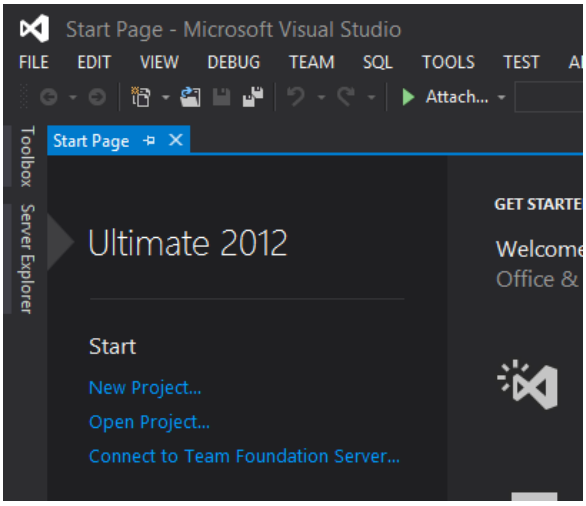
Figura 4.9: Pantalla de inicio de Visual Studio 2012 Express

Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena.

En la página de inicio, se selecciona la opción Nuevo Proyecto

Figura 4.10: Página de inicio de Visual Studio



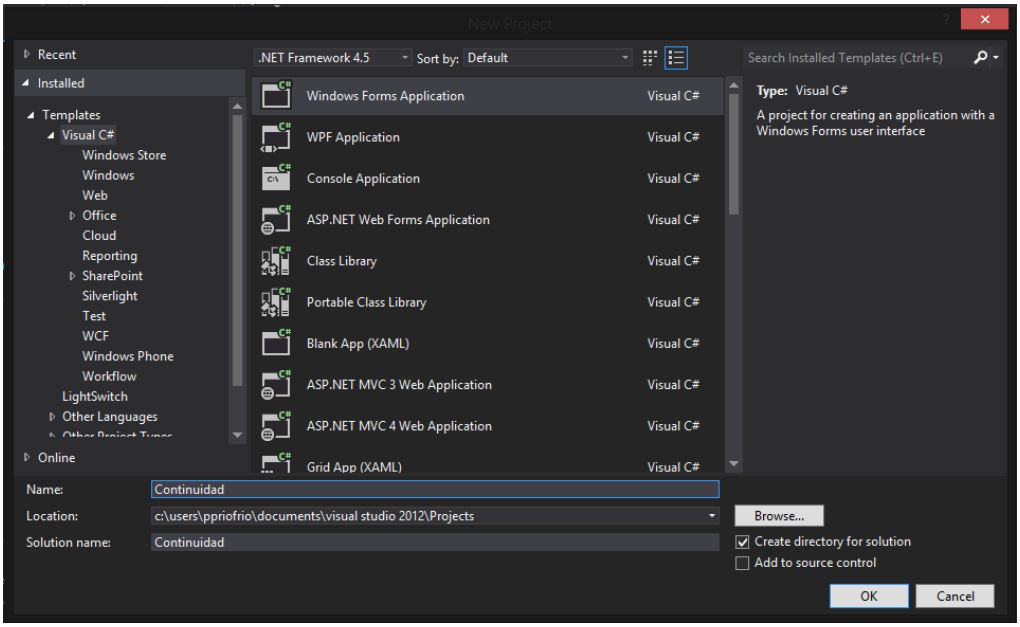
Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Selecciona el tipo de proyecto que se va a construir, en nuestro caso es una aplicación de Windows (Windows FormsApplication).

A continuación selecciona un nombre para la solución y dejamos marcada la opción Crear Directorio para la Solución. Clic en Aceptar.

Figura 4.11: Creación de la solución

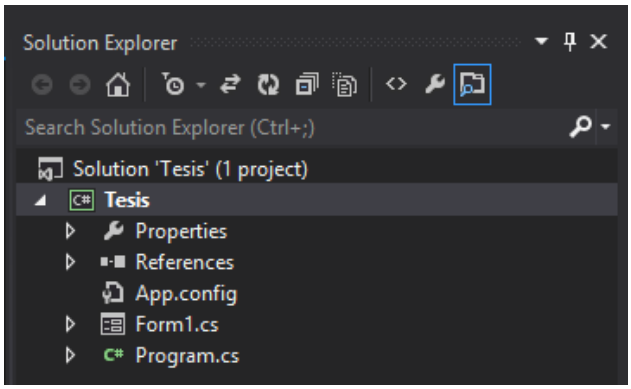


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

En el explorador de soluciones podemos ver que se ha creado un proyecto que contiene un formulario por defecto (Form1.cs) y el código necesario en el archivo Program.cs para que se lance dicho formulario al ejecutar la aplicación.

Figura 4.12: Vista de la solución en la ventana de explorador del proyecto



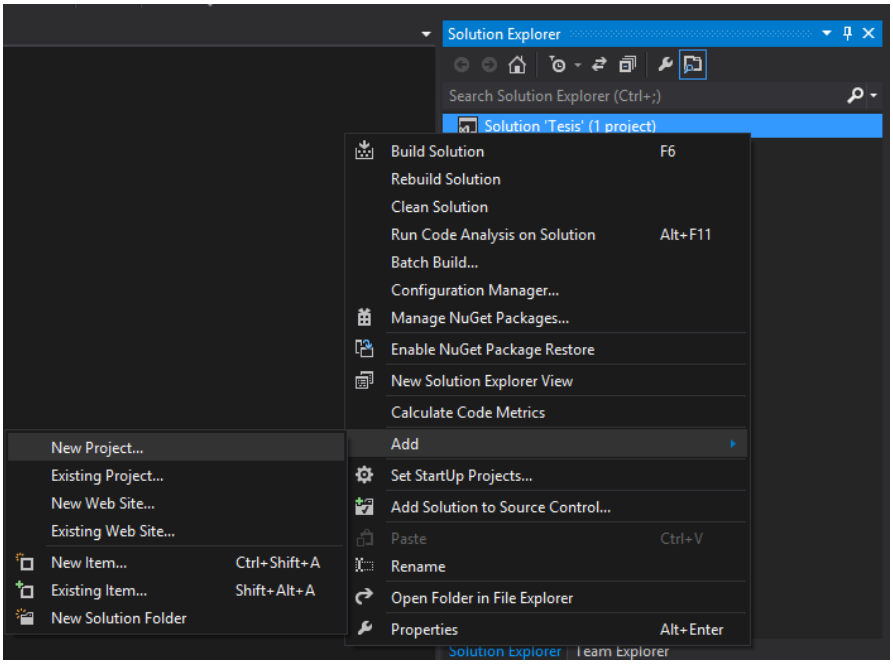
Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

El siguiente paso es dar clic derecho en el nombre del proyecto y seleccionar cambiar nombre. A este proyecto se lo llamara UI puesto que representa dicha capa de la arquitectura.

Luego damos clic derecho sobre la solución y selecciona Agregar – Nuevo Proyecto.

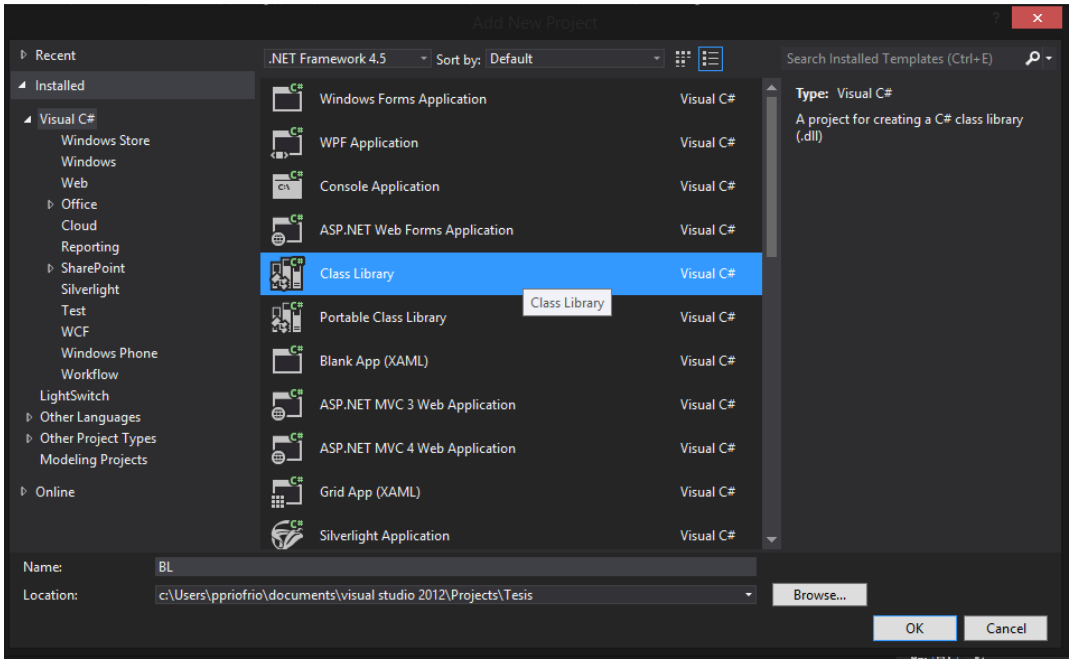
Figura 4.13: Agregar un proyecto a la solución



Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

En los parámetros del nuevo proyecto selecciona como tipo Librería de Clases y como nombre del proyecto BL, puesto que representará dicha capa.

Figura 4.14: Parámetros de creación del proyecto BL



Fuente: Visual Studio 2012 Express.

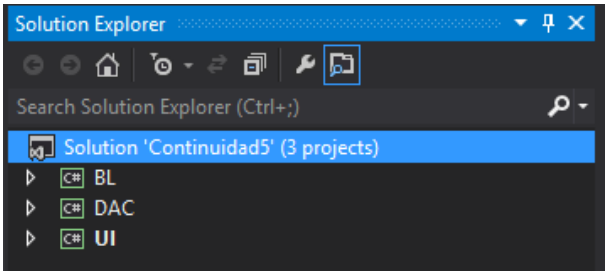
Autor: Alan Yamil Correa Requena

Luego agrega otro proyecto del mismo tipo y como nombre seleccionamos DAC porque contendrá la capa de acceso a datos.

Al crear estos proyectos se creará por defecto una clase (Class1). Se puede eliminarla sin problemas.

El esquema quedará compuesto por 3 proyectos con el nombre de la capa correspondiente.

Figura 4.15: Vista de la solución con las capas creadas

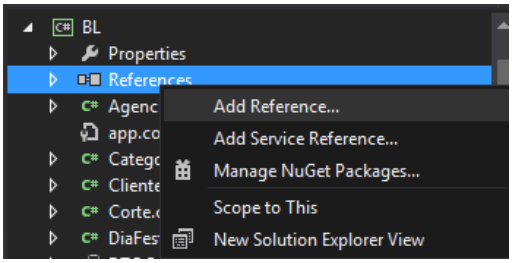


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Lo siguiente es agregar las referencias. Para ello hace clic derecho en el nodo References de los proyectos y manipula de tal manera que la UI referencie a la BL y la BL referencie a la DAC. Es decir, en la UI ponemos Add Reference y escogemos BL. En BL ponemos Add Reference y escogemos DAC, desde la pestaña Projects de la ventana Añadir Referencia.

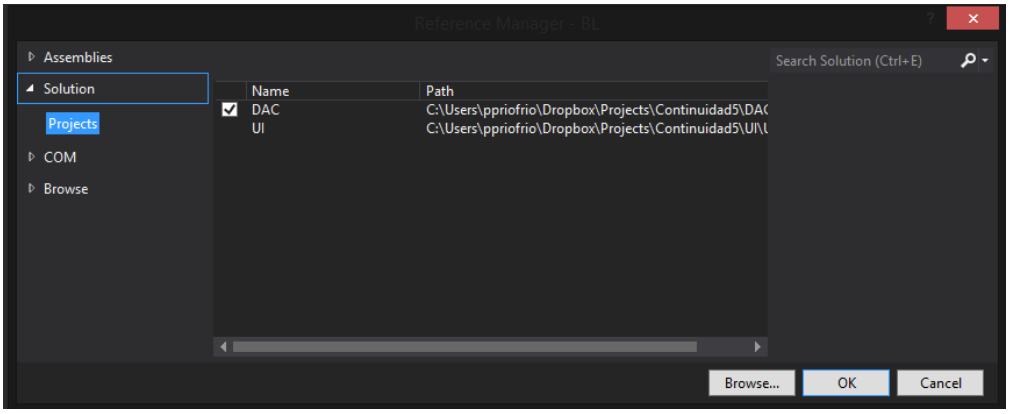
Figura 4.16: Agregar referencias al proyecto



Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Figura 4.17: Referencias del proyecto BL



Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

4.4.3.2. DAC

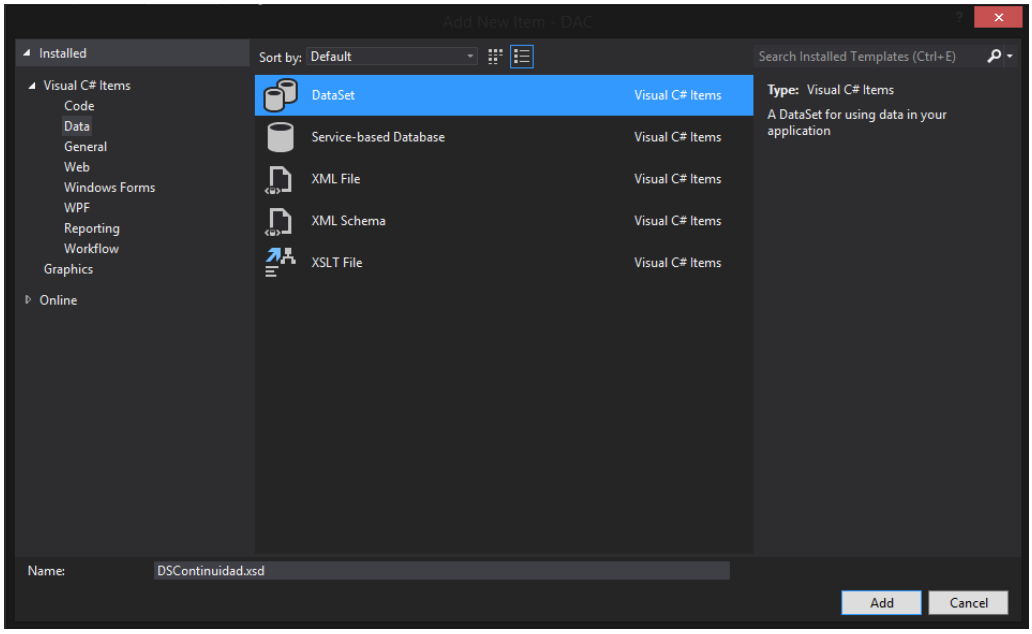
Creación de un DataSet y tableAdapters

Para conectar con la base de datos utilizaDataSets y TableAdapters. Los pasos para crearlos son los siguientes:

En el proyecto DAC, mediante el menú contextual Add->New Item, crear un nuevo elemento de tipo DataSet.

Ponemos el nombre del DataSet de acuerdo al paquete organizativo que va a representar. En este caso manejar un solo DataSet que representará a toda la base de datos.

Figura 4.18: Agregar un DataSet al proyecto

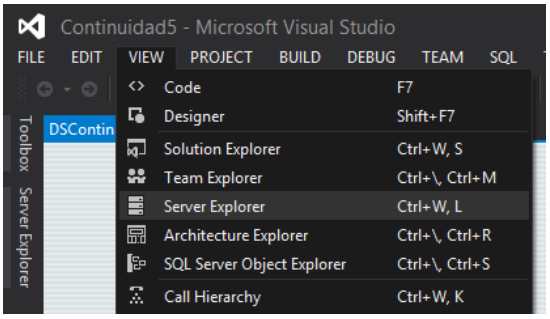


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Se ha creado un DataSetvacío, para poblarlo necesita asignarle las tablas que va a manejar. Si no está visible, mostrar la ventana del Explorador de Servidores.

Figura 4.19: Mostrar el Explorador de Servidores



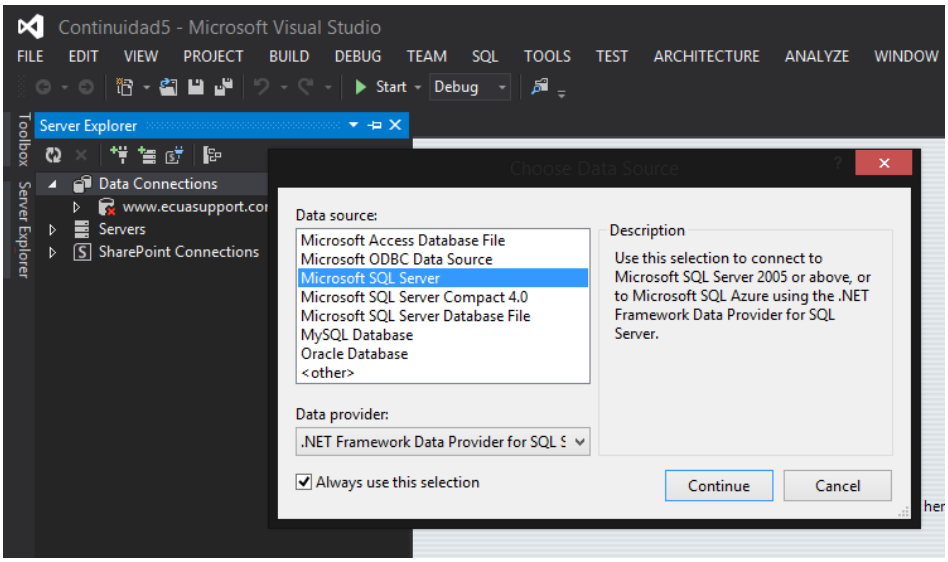
Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

En el explorador de servidores, presiona el botón Conectar a Base de Datos representado por un conector con una cruz verde. Esto nos llevará a la selección de una fuente de datos

o Data Source. En esta pantalla se listarán todos los conectores de base de datos que estén instalados en el equipo. Elegimos Microsoft SQL Server y como Data Provider, el .Net Framework para SQL Server.

Figura 4.20: Conectar a una base de datos

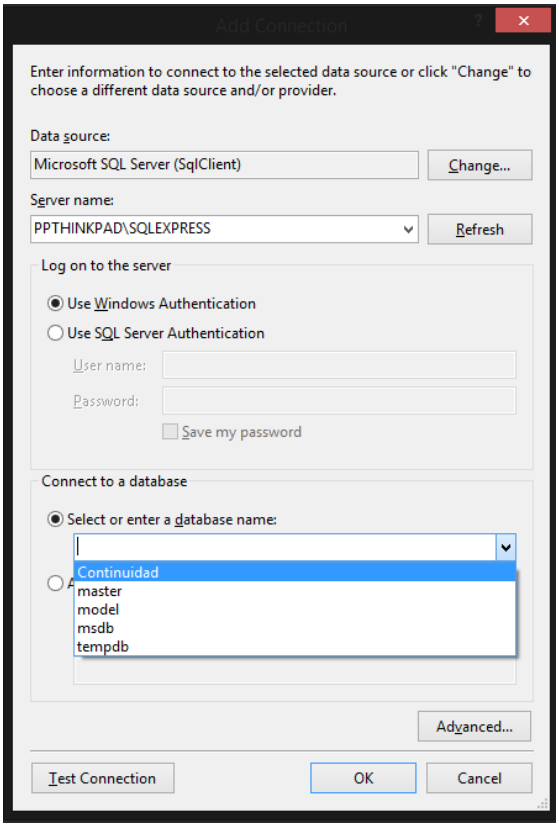


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Aparece un asistente donde debe configurar los datos de la conexión a BDD. Para nuestro proyecto quedaría así:

Figura 4.21: Configuración de la conexión

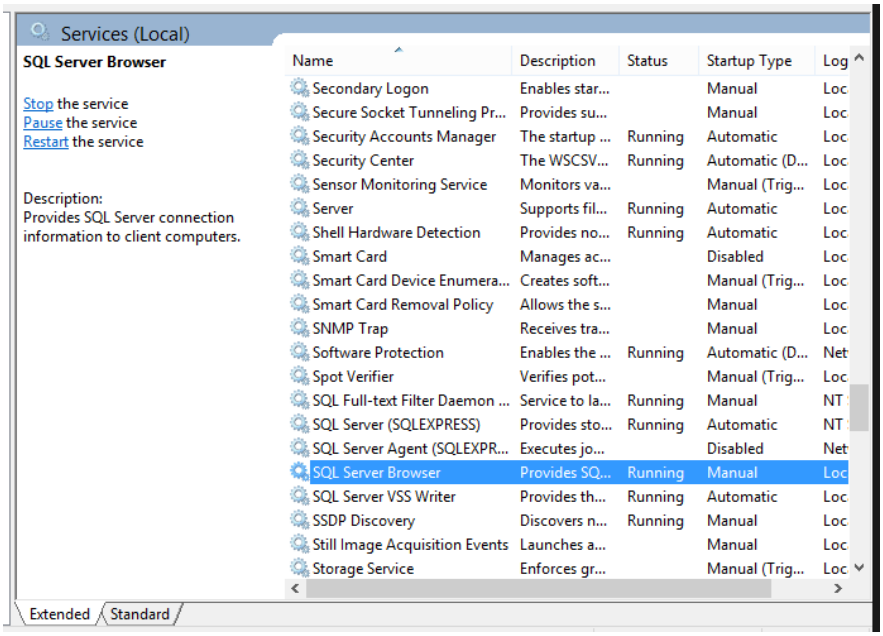


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

En el campo Server Name coloca el nombre del equipo o la dirección IP donde se esté ejecutando el servicio de Base de Datos. Si no aparece en la lista de servidores, probablemente no se esté ejecutando el servicio SQL Server Browser. Para comprobarlo y/o corregirlo va a los servicios de Windows (Windows + R, “services.msc”).

Figura 4.22: Servicios de Windows

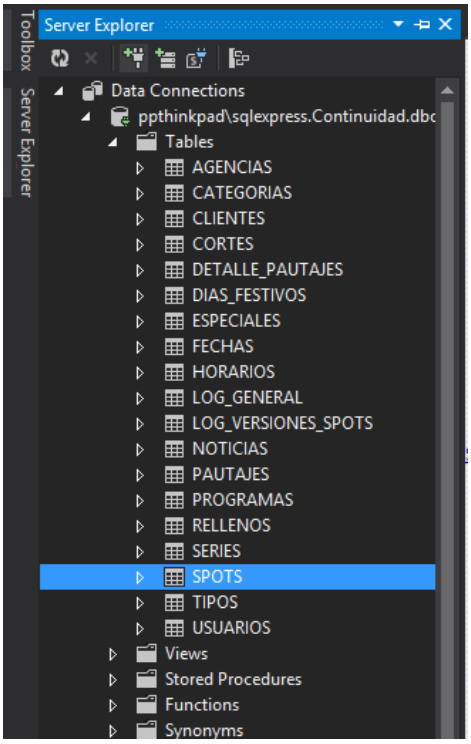


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Una vez realizada la conexión, en la ventana de Server Explorer aparecerá todo el esquema de la base de datos, desde donde puede arrastrar hacia el dataSet las tablas necesarias.

Figura 4.23: Vista expandida del Server Explorer

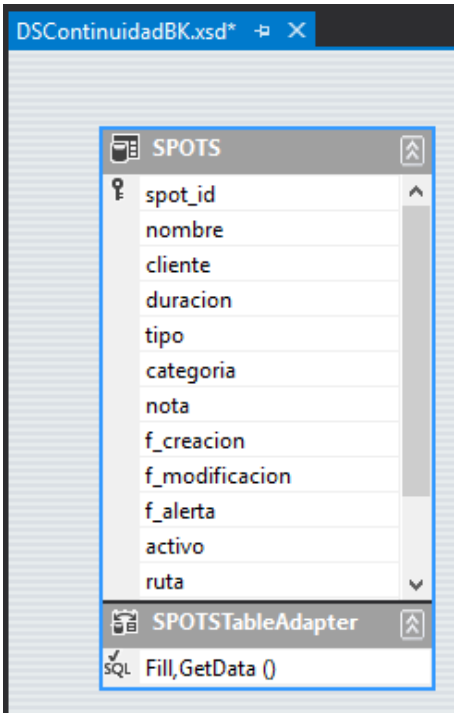


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Una vez arrastradas hacia el dataSet, se crean DataTables con sus correspondientes TableAdapters. Los DataTables representan la estructura de la tabla y los TableAdapters permiten la comunicación con la base de datos a través de consultas (Querys) y sentencias DML (Data ModifyingLanguage).

Figura 4.24: DataTable con TableAdapter



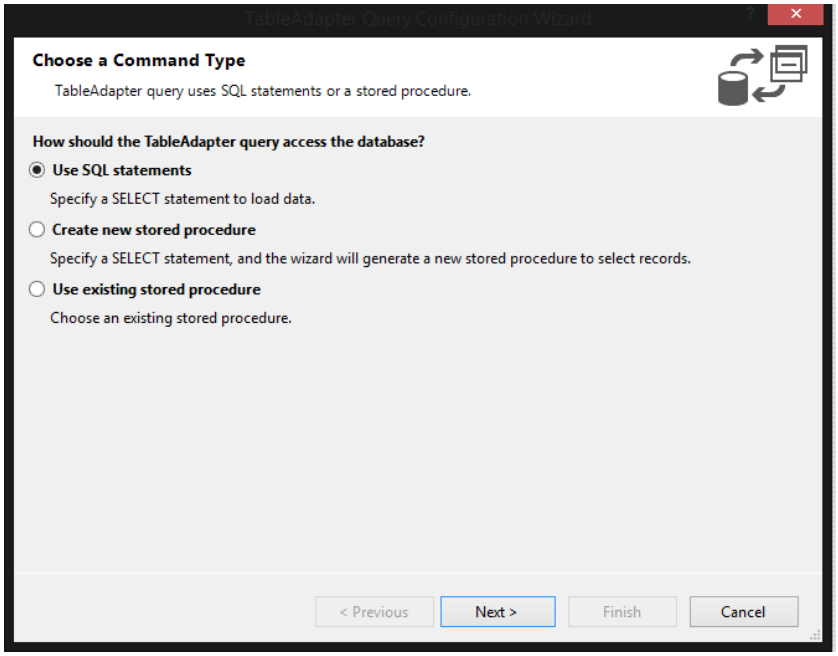
Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Los tableAdapters traen por defecto los métodos Fill, GetData, Insert, Update y Delete. Estos son generados en base a la estructura de la tabla. Si la tabla no tiene clave primaria, no se generan automáticamente las sentencias Update ni Delete.

Para crear una consulta personalizada, hacemos clic derecho sobre el TableAdapter y seleccionamos AddQuery. Existen dos tipos de comandos: SQL y Procedimiento Almacenado, el primer paso es escoger uno de ellos.

Figura 4.25: Asistente de configuración de consultas – tipo de comando

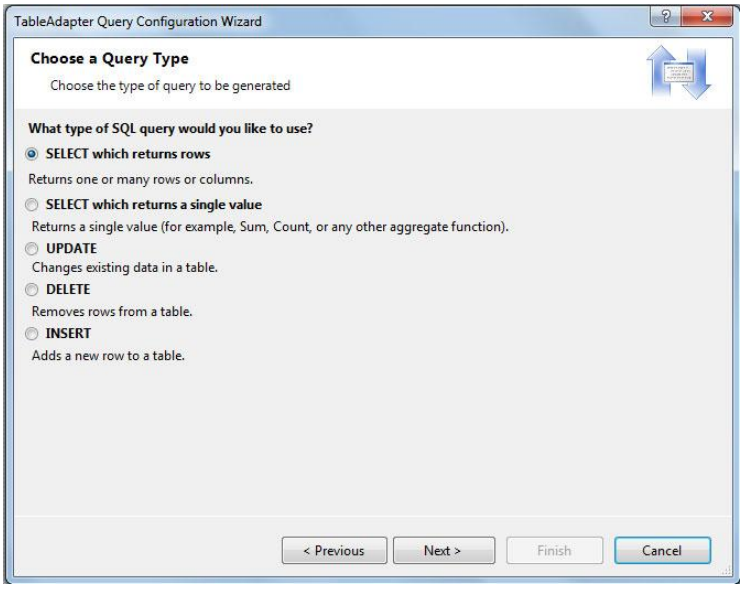


Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

El paso siguiente es elegir el tipo de comando. La primera opción devolverá una colección de filas (una tabla), la segunda nos retornará un valor único, esto usualmente se utiliza para recuperar algún subtotal o total usando comandos de agregación SQL como pueden ser SUM(), COUNT(), MAX(), MIN().

El resto de opciones se utilizan para crear distintas sentencias de Inserción y modificación de datos.

Figura 4.26: Asistente de configuración de consultas – tipo de consulta

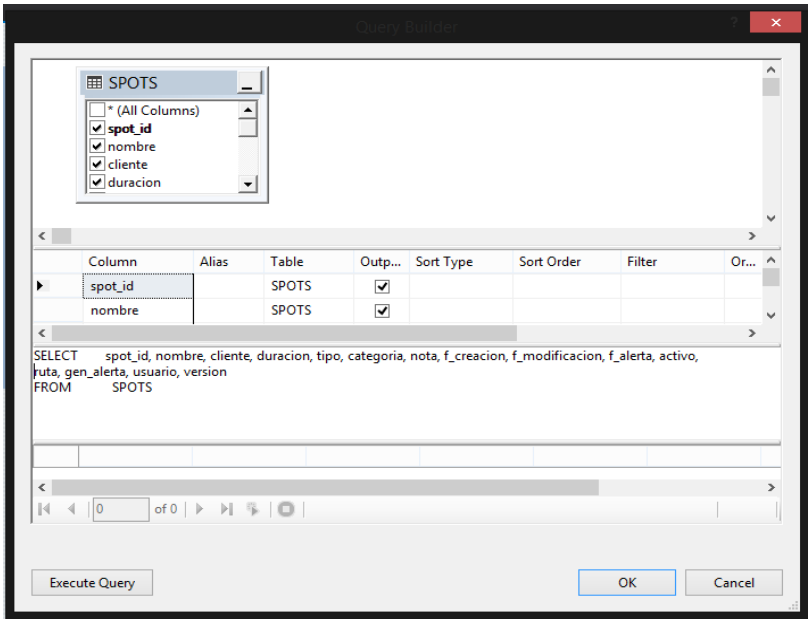


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Aparece una ventana donde debe configurar nuestra consulta, lo cual puede hacerse usando el QueryBuilder o escribiendo directamente el código SQL.

Figura 4.27: Constructor de consultas (QueryBuilder)



Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Una vez establecida la consulta mediante el QueryBuilder o escrita directamente sobre la ventana del editor, el asistente nos da a escoger si genera métodos Fill, GetData o ambos. Los métodos Fill llenan el DataTable con los resultados de la consulta y devuelven un número entero que representa la cantidad de filas afectadas.

Los métodos GetData no llenan el DataTable, solo devuelven una nueva tabla con los resultados de la consulta. Los GetData pueden devolver una tabla con estructura diferente del DataTable que contiene el TableAdapter, los métodos Fill deben ajustarse a la estructura establecida.

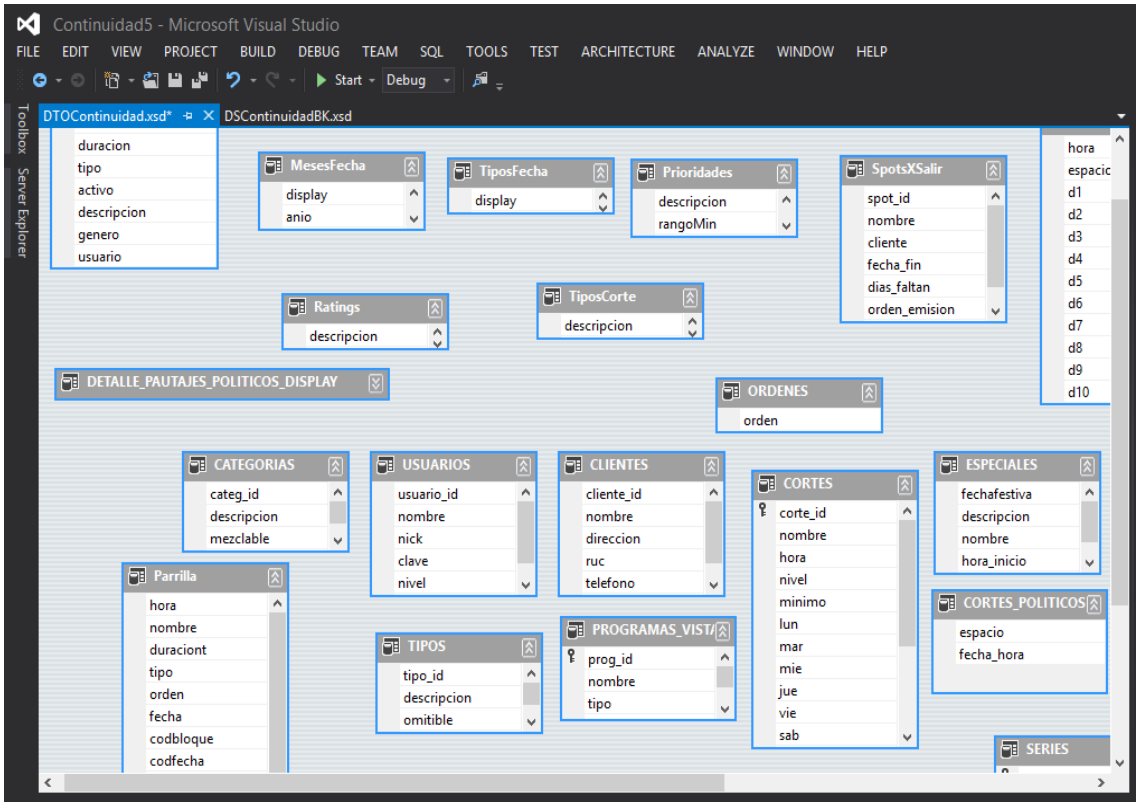
4.4.3.3. BL

Construcción de DTOs

Los DTO son objetos de transporte de datos, que no son más que DataSets que contienen DataTables sin TableAdapters, y cuya estructura es la misma que la de los DataSets de la capa DAC.

Para crear el DTO, lo que debe hacer es crear un nuevoDataSet en la capa BL, copia del DSContinuidad de la DAC todos los DataTables y los pega (Ctrl+V) en el DataSet de la BL, al cual previamente he nombrado DTOContinuidad.

Figura 4.28: DataSet de transporte (DTO)



Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

Una vez pegados, selecciona los tableAdapters y los elimina con la tecla DEL o Supr del teclado. Tiene listo el DTO.

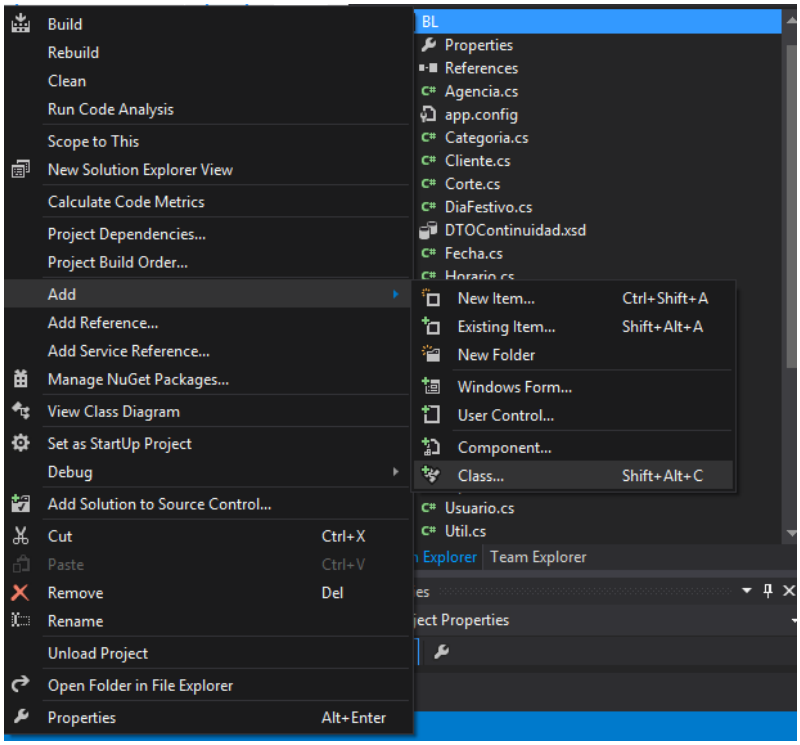
En el DTO se definen también tablas auxiliares que son llenadas en tiempo de ejecución y sirven como contenedores de datos. Estas tablas auxiliares no necesitan estar enlazadas a la base de datos sino que únicamente sirven para organizar colecciones de datos tipificados y resultan más fáciles de utilizar que las colecciones estándar que provee .NET.

Construcción de clases

Para la construcción de una clase se siguen los siguientes pasos:

En el proyecto BL, en la carpeta correspondiente, da clic derecho, Agregar Nuevo ítem y selecciona Clase (Class).

Figura 4.29: Agregar una clase

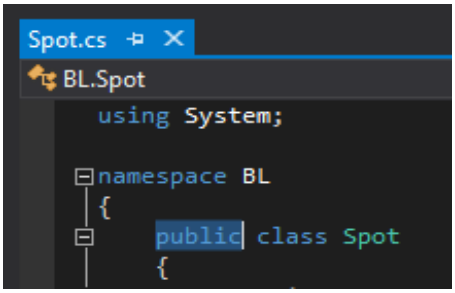


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

En este caso va a construir la clase Spot. Al poner Añadir, se mostrará en la ventana del Editor el código inicial de la clase. Por defecto el nivel de acceso es privado. Para poder usar la clase en otro proyecto o capa, debe definirla como clase pública de la siguiente manera.

Figura 4.30: Accesibilidad de la clase



Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Luego crea los objetos de DTO (Data transportobject) y los tableAdapters. Los objetos DataTable y DataRow (dt y dr) se definen como propiedades (Alt+R+E en Visual Studio genera automáticamente los getters y setters). En el constructor de la clase se instancian los objetos creados. El código queda de la siguiente manera.

```
publicclassSpot
   {
      # region DECLARACION

DTOContinuidad.SPOTSDataTabledt;
publicDTOContinuidad.SPOTSDataTableDt
      {
get { returndt; }
set { dt = value; }
      }

DTOContinuidad.SPOTSRowdr;
publicDTOContinuidad.SPOTSRowDr
```

```
        {
get { returndr; }
set { dr = value; }
}

DAC.DSContinuidadTableAdapters.SPOTSTableAdapterta;

        # endregion

        # region INICIALIZACION

public Spot()
        {
ta = newDAC.DSContinuidadTableAdapters.SPOTSTableAdapter();
}       }
```

Nótese que los objetos dt y dr son instanciados desde la BL y los tableAdapters son instanciados desde la DAC. El objeto ta se manejará internamente dentro de la clase mientras que los dt y dr podrán ser accesados desde la UI.

Construcción de Métodos

La mayoría de métodos de las clases del proyecto son de acceso público porque se llaman desde la capa de Interfaz de usuario o desde otras clases de la BL.

Para los métodos de obtención de datos se usa un código donde primero se comprueba si se ha inicializado la tabla, si no se ha hecho, entonces la inicializa. Si ya ha sido inicializada, la vacía a través del método Clear.Se recurre al método Merge para pasar los datos que devuelve el tableAdapter desde la DAC al DTO de la BL, esto por las restricciones de la arquitectura que impiden que la Interfaz de Usuario acceda directamente a la base de datos.

```
publicvoidInicializarDt()
        {
if (dt == null) dt = newDTOContinuidad.SPOTSDataTable();
elseif (dt.Count> 0) dt.Clear();
}
publicintLlenarTodos()
        {
InicializarDt();

try { dt.Merge(ta.GetData()); }
catch { throw; }

returndt.Count;
}
```

Nótese que los métodos de este tipo devuelven un número entero que es el número de registros de la tabla afectada. Este es el estándar que se usará para todos los métodos

similares. Todos los métodos incluyen bloques try catch. Si se produce una excepción, esta es relanzada para ser capturada y presentada como un error en la capa UI.

Para los métodos de inserción se procede de la siguiente forma:

```
publicintInsertar(inti)
        {
intresultado = 0;

try
            {
resultado = ta.Insert(dt[i].nombre, dt[i].cliente, (int)dt[i].duracion, dt[i].tipo,
dt[i].categoria, dt[i].nota,
dt[i].f_creacion, dt[i].f_modificacion, dt[i].f_alerta, dt[i].activo, dt[i].ruta,
dt[i].gen_alerta, dt[i].usuario, dt[i].version);
}
catch { throw; }

return resultado;
        }
```

Se recibe como parámetro únicamente la posición del BindingSource que se maneja en la UI. Puesto que en esa capa es donde se encuentran enlazados los controles a la tabla dt de la BL a través de dicho BindingSource, los cambios o inserciones que realice el usuario son reflejados instantáneamente en la tabla Dt.

Con estos datos se llama al método Insert generado en el TableAdapterta, el cual devuelve el número de filas afectadas en la base de datos.

Para los métodos de modificación o actualización el código es el siguiente:

```
publicintModificar(inti)
{
dt[i].BeginEdit();
dt[i].f_modificacion = System.DateTime.Now;
dt[i].EndEdit();

intresultado = 0;

try { resultado = ta.Update(dt[i].nombre, dt[i].cliente, dt[i].duracion, dt[i].tipo,
dt[i].categoria, dt[i].nota,
dt[i].f_modificacion, dt[i].activo, dt[i].ruta, dt[i].gen_alerta, dt[i].usuario, dt[i].version,
dt[i].spot_id); }
catch { throw; }

return resultado;
    }
```

Igualmente se recibe como parámetro la posición del BindingSource. El método Update del tableAdapter devuelve el número de filas afectadas en la base de datos.

4.4.3.4. UI

Construcción de un formulario

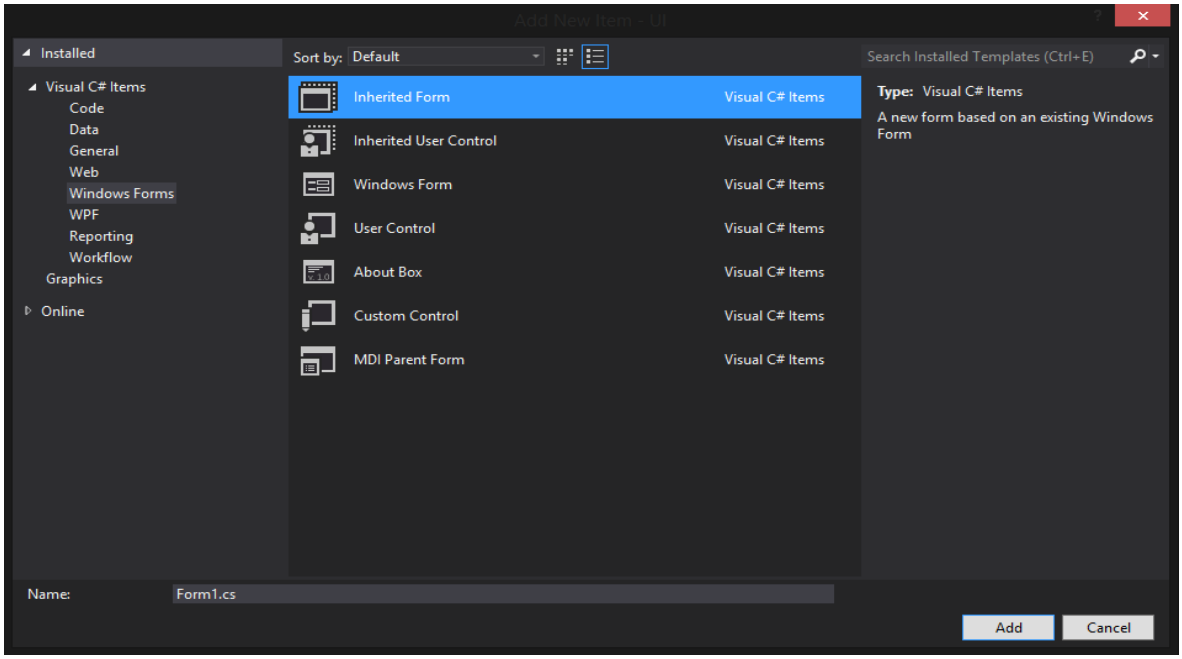
Los formularios de la aplicación son heredados de un formulario personalizado llamado BaseForm, el cual incluye métodos virtuales que pueden ser sobrescritos por los formularios hijos, así como atributos y métodos comunes que agilizan la construcción de los formularios heredados haciendo uso del poder de la programación orientada a objetos.

Para crear un formulario se debe seguir los siguientes pasos:

En el proyecto UI, da clic derecho, Agregar Nuevo ítem.

Escoge a la izquierda Windows Forms, luego al centro, InheritedForm (Formulario Heredado), damos un nombre al formulario y aceptamos.

Figura 4.31: Agregar un formulario heredado

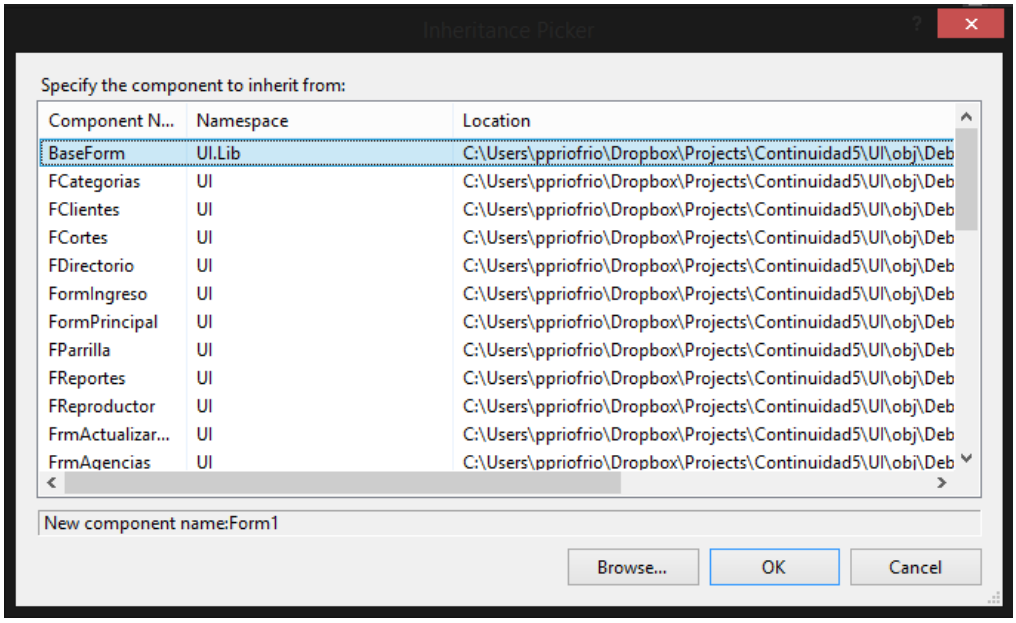


Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

Se presentará una pantalla donde podemos escoger la clase base de la que se va a heredar.

Figura 4.32: Elegir la clase base para la herencia del formulario



Fuente: Visual Studio 2012 Express.

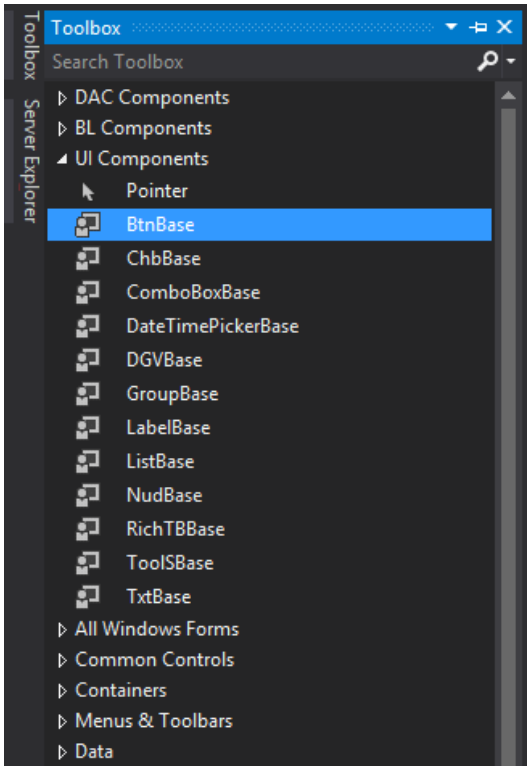
Autor: Alan Yamil Correa Requena

EscogeBaseForm del NamespaceUI.Lib y aceptamos.

Se presenta un formulario en blanco que posee todas las propiedades y métodos de la clase BaseForm. Ahora procedemos a agregar controles según sea necesario, ya sea los convencionales de Visual Studio o los componentes personalizados que se han desarrollado para el presente proyecto.

Para agregar controles al formulario podemos arrastrarlos desde la ventana ToolBox hacia el formulario.

Figura 4.33: Caja de herramientas – controles personalizados



Fuente: Visual Studio 2012 Express.

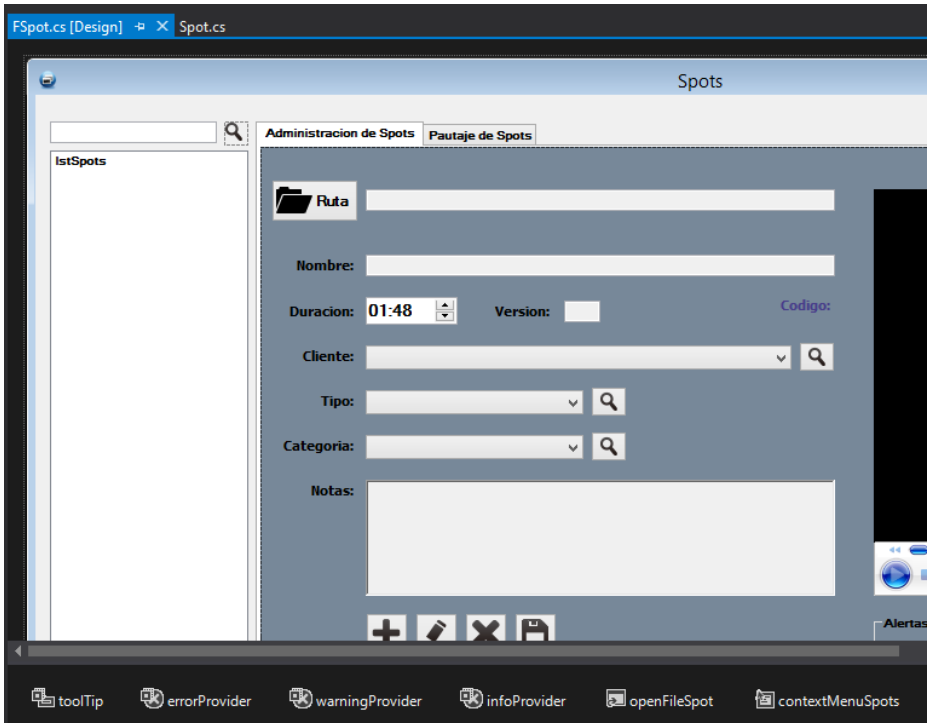
Autor: Alan Yamil Correa Requena

En este ejemplo he insertado en el formulario un ListBoxBase, varios BtnBase, algunos ComboBoxBase desde la librería de controles personalizados, y desde la librería estándar de Visual Studio, algunos Label y un TabControl. Puede observar en la parte inferior que son parte del formulario algunos componentes heredados del formulario base como son el toolTip, el errorProvider, el warningProvider y el infoProvider.

Además, desde la librería de componentes COM, se ha agregado el control ActiveX de Windows Media Player, el cual nos servirá para visualizar y analizar el contenido multimedia.

Desde la ventana de Propiedades puede ajustar y revisar la mayoría de parámetros y ajustes del formulario. En esta ventana hemos configurado la propiedad Text del formulario para que diga “Spots”, los nombres de los controles y algunas propiedades de cada control.

Figura 4.34: Diseño del formulario Spots

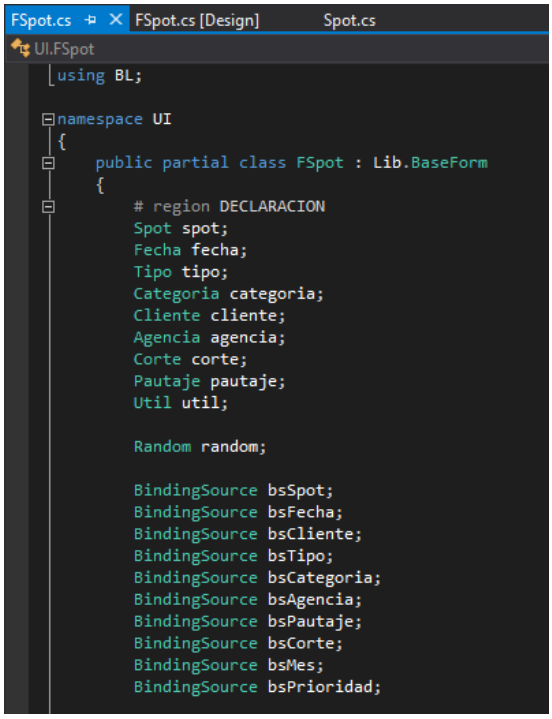


Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

Al presionar Ctrl + Alt + Enter pasa a la vista de código del formulario donde va a definir la funcionalidad del mismo.

Primero se crean los objetos de las clases de la BL que se van a utilizar en el formulario y los BindingSource que van a vincular los datos con los controles.

Figura 4.35: Vista de código del formulario Spots



Fuente: Visual Studio 2012 Express.

Autor: Alan Yamil Correa Requena

En el constructor inicializa los objetos y llama al método EnlazarControles.

```
publicFSpot()
        {
spot = newSpot();
fecha = newFecha();
tipo = newTipo();
categoria = newCategoria();
agencia = newAgencia();
cliente = newCliente();
corte = newCorte();
pautaje = newPautaje();
util = newUtil();
```

```
random = newRandom();

bsSpot = newBindingSource();

bsFecha = newBindingSource();

bsCliente = newBindingSource();

bsTipo = newBindingSource();

bsCategoria = newBindingSource();

bsAgencia = newBindingSource();

bsPautaje = newBindingSource();

bsMes = newBindingSource();

bsCorte = newBindingSource();

bsPrioridad = newBindingSource();

bufferPautaje = newDTOContinuidad.PAUTAJESDataTable();


        limite1 = 1;

        limite2 = 28;

primerDia = 1;

cursorSpot = -1;


Parametroparametro = newParametro();

filtroSpot = parametro.InicializarFiltroSpots();

filtroCorte = parametro.InicializarFiltroCortes();

filtroCorte.lunesAviernes = 1;


InitializeComponent();
```

```
EnlazarControles();
    }
```

En el método EnlazarControles primero inicializa los DataTables del objeto de la BL (Spot), luego asigna al BindingSourcebsSpots dicho DataTable como fuente de datos. Ahora el bsSpots puede a su vez ser fuente de datos de los controles como el listBoxlstSpots, puesto que se ha creado una vinculación entre el bsSpots y los datos.

```
privatevoidEnlazarControles()
    {
spot.Dt = newDTOContinuidad.SPOTSDataTable();
bsSpot.DataSource = spot.Dt;
lstSpots.DataSource = bsSpot;
lstSpots.DisplayMember = spot.Dt.nombreColumn.ColumnName;
lstSpots.ValueMember = spot.Dt.spot_idColumn.ColumnName;
```

Los controles tienen una propiedad denominada DataBindings, a la cual podemos asignar una fuente de datos y un miembro de datos para que mantengan el vínculo entre sus propiedades y los datos.

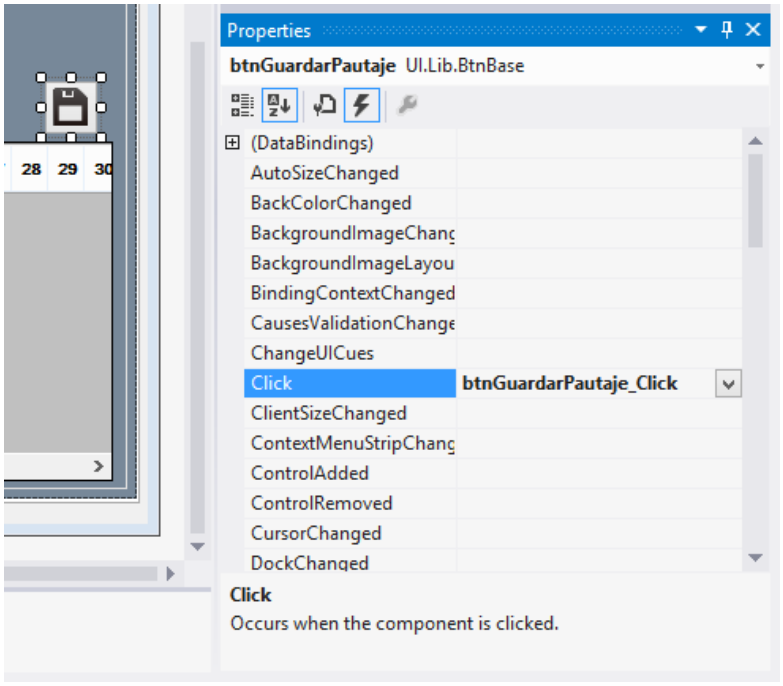
En el caso de los controles TextBox, el DataBinding que asigna tiene los siguientes parámetros: propiedad del control que va a enlazarse, fuente de datos y nombre del miembro de datos vinculado.

```
txtRuta.DataBindText(bsSpot, spot.Dt.rutaColumn.ColumnName);
```

txtNombreSpot.DataBindText(bsSpot, spot.Dt.nombreColumn.ColumnName);

Para definir la funcionalidad de los controles debe crear eventos. Los eventos puede crearlos a través de líneas de código o usando la interfaz de Visual Studio. Para definir un evento usando la interfaz, selecciona el control y en la ventana de propiedades filtra la sección eventos, localiza el evento deseado y damos doble clic.

Figura 4.36: Vista de los eventos de los controles



Fuente: Visual Studio 2012 Express.
Autor: Alan Yamil Correa Requena

Esto nos lleva a la vista código donde puede definir la acción a realizar cuando se ejecute el evento en cuestión. Es aconsejable llamar a un método en lugar de escribir directamente el código en el evento.

```
privatevoidbtnGuardarPautaje_Click(objectsender, EventArgs e)
    {
```

```
Guardar(gpPautaje);
dgCortes.Focus();
    }
```

Los métodos que se deben programar en los formularios pueden ser overrides de los métodos del BaseForm o métodos nuevos. Se hace un override en el caso de algunos eventos y de los métodos enlazados a la barra de herramientas del menú principal (Insertar, Editar, Guardar, etc.)

Para sobrescribir por ejemplo el método InsercionIniciar (asociado al botón nuevo de la barra de herramientas) se define un método público con la palabra clave override seguida del nombre del método a sobrescribir. De forma automática Visual Studio inserta las líneas de código que llaman a la funcionalidad base del método, dígase base.InsercionIniciar (que podemos omitir o reubicar). Adicionalmente se escribe nuestro propio código que debe incluir las líneas necesarias para iniciar el proceso de inserción en el formulario y en el objeto spot (BL).

```
publicoverridevoidInsercionIniciar(ControlpContenedor)
    {
lstSpots.Enabled = false;

if (pContenedor.Name == "tabSpots")
        {
cursorSpot = bsSpot.Position;
```

```
spot.Dt.AddSPOTSRow(spot.GetMaxCod() + 1, "", 0, 0, 0, 0, "", DateTime.Today,
DateTime.Now, DateTime.Today, 1, "", 0, Properties.Settings.Default.usuario.Dr.nick,
"A");
bsSpot.Position = spot.Dt.Count - 1;
btnRuta.Select();

AcceptButton = btnRuta;
        }

base.InsercionIniciar(pContenedor);
    }
```

En los métodos de inserción se agrega una fila más al DataTable del objeto spot en este caso mediante el método spot.Dt.AddSPOTSRow. Como los controles están vinculados al DataTable, si escribe en los campos de texto o cambia la selección en los combos, simultáneamente esta cambiando los datos en la fila que acabamos de añadir.

Si cancela la inserción antes de guardar, se elimina la fila agregada mediante el método

Dt.RejectChanges o Dt.RemoveCurrent.

```
publicoverridevoidInsercionCancelar(ControlpContenedor)
    {
lstSpots.Enabled = true;
spot.Dt.RejectChanges();
```

```
lstSpots.Focus();

AcceptButton = null;

base.InsercionCancelar(pContenedor);
    }
```

El método Guardar primero llama al método Validar donde se verifica que los datos estén completos y sean coherentes. Este método retorna un valor booleano que es verdadero si los datos han sido validado, caso contrario se llama al errorProvider incluido en el BaseForm para que muestre en pantalla una advertencia o señal de error junto al control que no ha sido validado.

```
privateboolValidarSpot()
    {
bool vale = true;

if (string.IsNullOrEmpty(txtRuta.Text))
{
vale = false;
errorProvider.SetError(txtRuta, "Campo obligatorio");
}

if (string.IsNullOrEmpty(txtNombreSpot.Text))
{
```

```
vale = false;
errorProvider.SetError(txtNombreSpot, "Campo obligatorio");
}

if (chkAlerta.Checked&&dtpFechaAlerta.Value<DateTime.Today)
{
vale = false;
errorProvider.SetError(dtpFechaAlerta, "Fecha de alerta no puede ser anterior a la fecha
actual");
}

if (spot.Dt[bsSpot.Position].duracion<= 0)
{
vale = false;
errorProvider.SetError(dtpDuracion, "Duracion debe ser mayor a 0");
        }

return vale;
    }
```

Si se validan los datos el método Guardar se refiere a las flags o variables de control heredadas del BaseForm y que son manejadas por las bases de los métodos sobrescribibles para determinar si se está realizando una inserción o una modificación a los datos. Según eso, el método Guardar llama al método correspondiente del objeto spot (BL), el cual

retornará un valor numérico que indica el número de registros afectados. El formulario presenta entonces un mensaje al usuario basado en este valor de retorno.

```
publicoverridevoidGuardar(ControlpContenedor)
      {
if (ValidarSpot())
            {
intresultado = 0;

if (_isInserting)
                {
try { resultado = spot.Insertar(bsSpot.Position); }
catch (Exception e)
                   {NotificarError(e.Message, "Error al insertar spot");
                   }
                }

elseif (_isEditing)
                {
try { resultado = spot.Modificar(bsSpot.Position); }
catch (Exception e)
{NotificarError(e.Message, "Error al modificar spot");
}
                }
```

```
if (resultado> 0)
                {
lstSpots.Enabled = true;
gpPautaje.Enabled = true;
lstSpots.Focus();
spot.Dt.AcceptChanges();

NotificarInformacion("Datos guardados satisfactoriamente");

base.Guardar(pContenedor);

AcceptButton = null;
                }
            }
elseNotificarAdvertencia("Datos no validados. Revise mensajes de error");
```

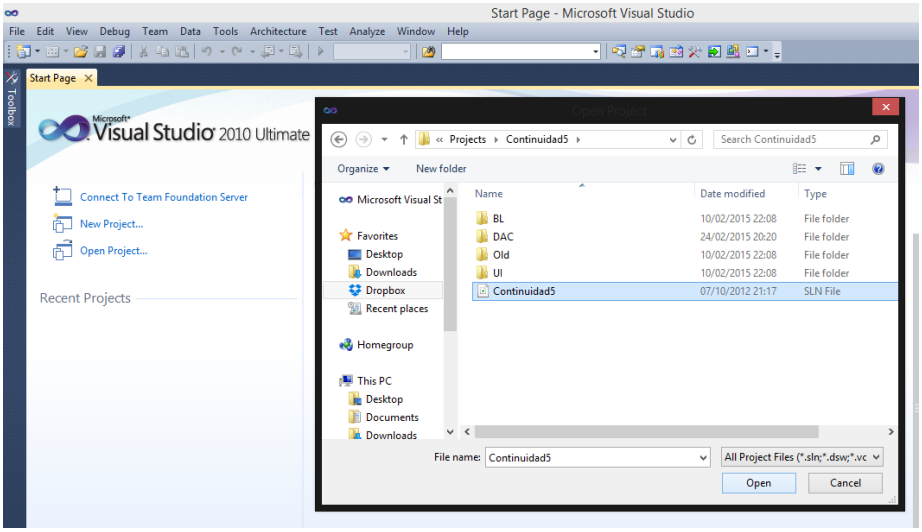
4.4.4. Reportes

Los reportes del proyecto se crearon con CrystalReports versión 13. Desafortunadamente a la fecha no existe una versión de esta herramienta que se acople al IDE de Visual Studio versión 2012 de ninguna edición, por lo que fue necesario recurrir a una instalación de evaluación de Visual Studio 2010, el cual si cuenta con soporte para CrystalReports.

4.4.4.1. Construcción de un reporte

Como primer paso abrir Visual Studio 2010 y recuperamos el proyecto.

Figura 4.37: Pantalla de inicio de Visual Studio 2010

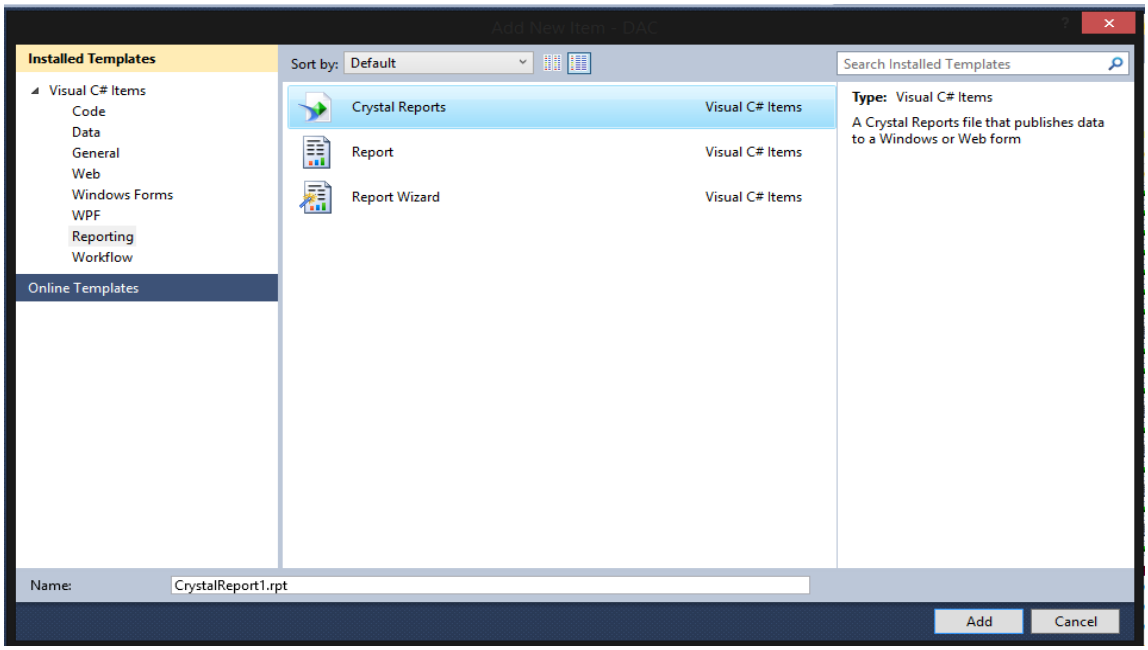


Fuente: Visual Studio 2010.

Autor: Alan Yamil Correa Requena

En la capa DAC da clic derecho y selecciona Agregar Nuevo Item. En la pantalla emergente selecciona a la izquierda Reporting y en el centro CrystalReports.

Figura 4.38: Agregar un elemento CrystalReports al proyecto



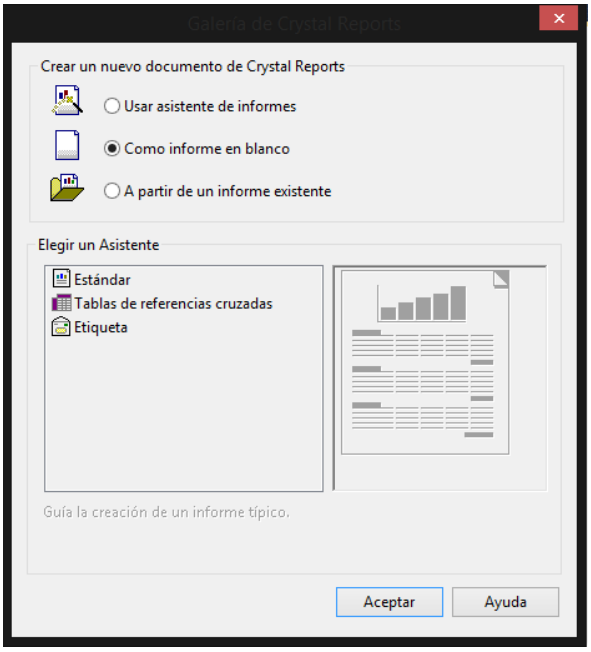
Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Asigna un nombre al reporte y damos clic en Add.

Se presenta una ventana donde se pregunta al usuario si desea utilizar un asistente o empezar un informe en blanco.

Figura 4.39: Galería de CrystalReports



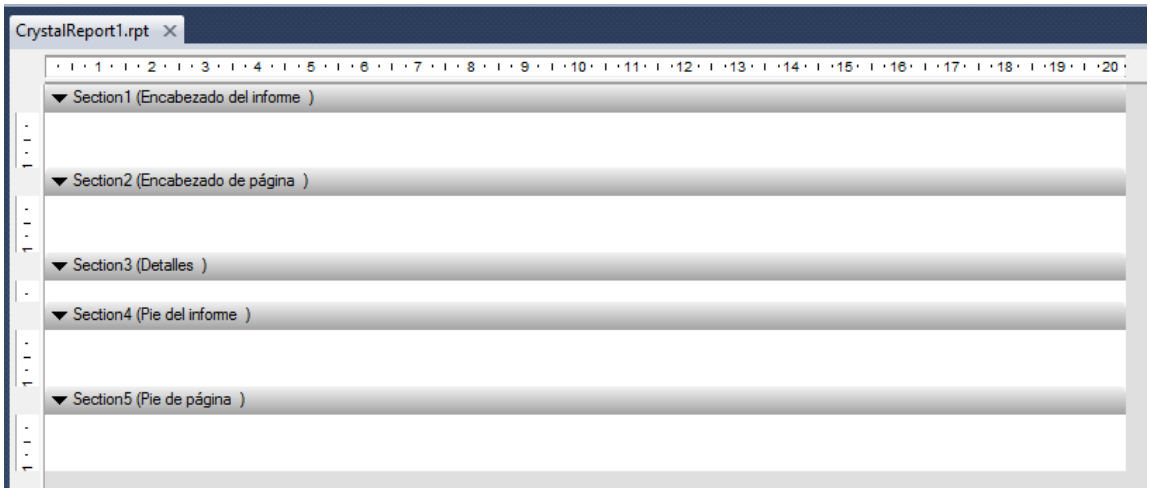
Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

En este caso elegir la opción de informe en blanco y damos Aceptar.

Se presenta la interfaz de diseño del CrystalReports. Por un lado tiene el lienzo o canvas que representa la hoja donde se plasmará el reporte. Por defecto incluye 5 secciones como se puede ver en la figura.

Figura 4.40: Secciones del reporte

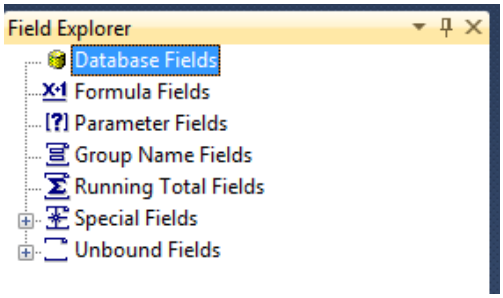


Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Por otro lado tiene el explorador de campos, desde donde puede seleccionar los datos, crear fórmulas, subtotales y agrupar los datos.

Figura 4.41: Field Explorer

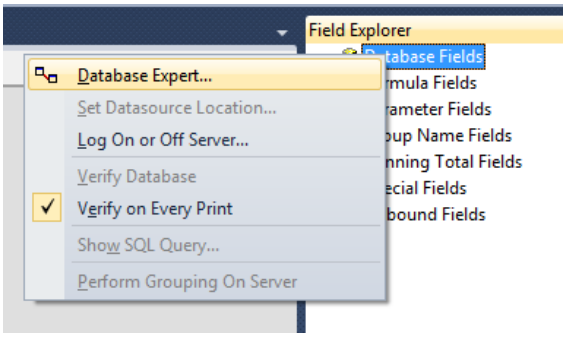


Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

En el explorador de campos da clic derecho sobre DatabaseFields y elege DatabaseExpert.

Figura 4.42: Llamar al DatabaseExpert

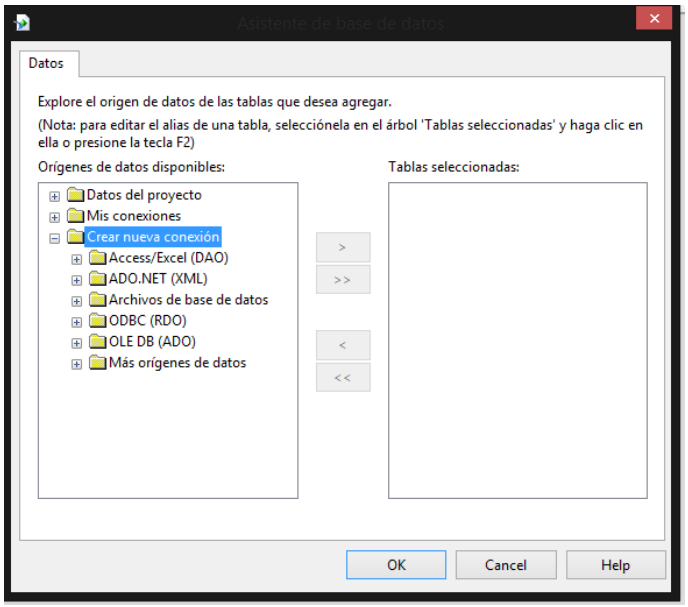


Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Se presenta una ventana donde podemos escoger el origen de datos para el reporte. Selecciona Crear nueva conexión y del árbol que se despliega escoge ADO.NET puesto que el origen de datos será un DataSet.

Figura 4.43: Asistente de base de datos de CrystalReports



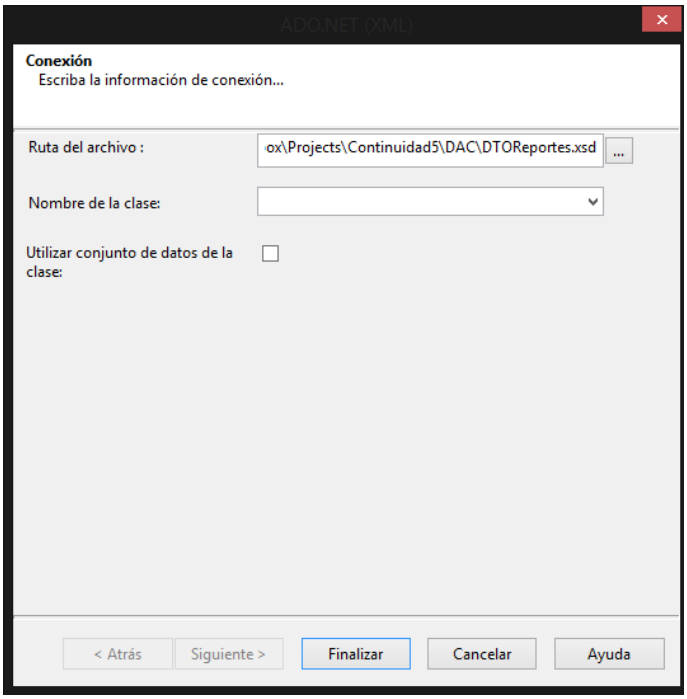
Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

En el cuadro que aparece debe introducir la ruta del DataSet que contiene la estructura de datos del reporte. Este DataSet se pasará como parámetro cuando se ejecute el reporte.

Por el momento nos interesa solamente la estructura. Nótese que la extensión del archivo es .xsd.

Figura 4.44: Selección del esquema XML

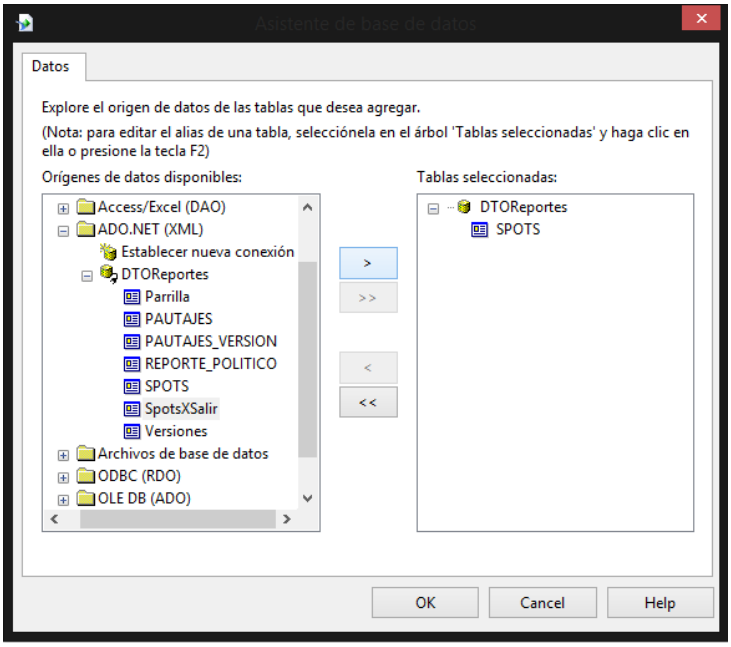


Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Ahora puede seleccionar las tablas que serán parte del reporte. Para ello puede usar el doble clic o el botón >. Una vez seleccionadas las tablas, damos clic en Aceptar. Si se seleccionan varias tablas podemos establecer vínculos que actuarán de forma similar a las claves foráneas en una base de datos.

Figura 4.45: Asistente de base de datos – elegir tablas

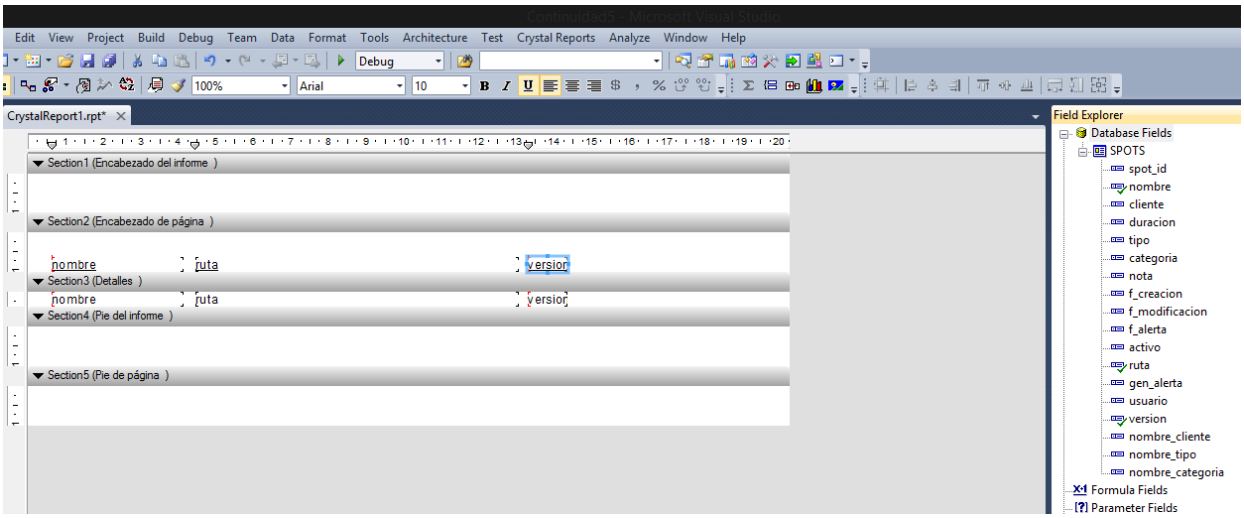


Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Ahora puede diseñar el reporte arrastrando los campos de las tablas hacia el lienzo.

Figura 4.46: Colocación de campos en el reporte



Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

Cuando un campo está utilizado en el reporte, se muestra con una marca verde en el explorador de campos.

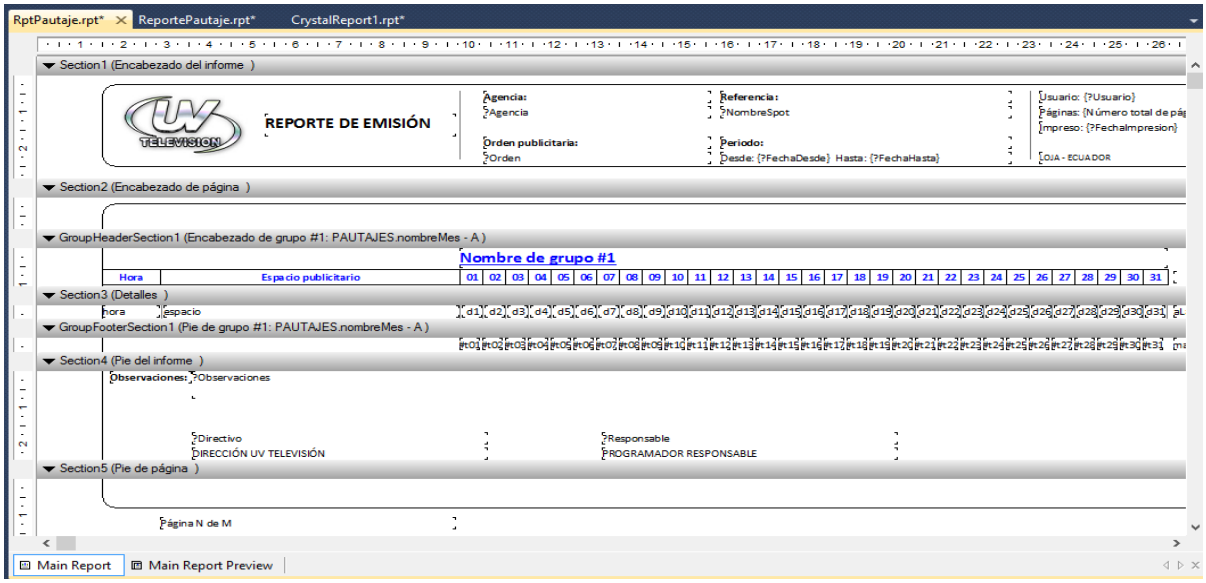
Dependiendo de la sección donde se coloquen los campos cambiará el comportamiento de los datos. Por ejemplo, si se colocan en el encabezado de página, los datos aparecerán una vez por cada nueva página; si se colocan en la sección Detalles, se creará una línea de detalle por cada registro que haya en la fuente de datos.

Los campos de fórmula y subtotales también dependen de la sección donde sean colocados.

Es posible agregar secciones y establecer condiciones de paginación, presentación, agrupamiento, etc. para lograr el resultado deseado.

Ejemplo de reporte completo:

Figura 4.47: Reporte de emisión de spot



Fuente: CrystalReports.

Autor: Alan Yamil Correa Requena

5. Pruebas de Software

5.1. Pruebas de software

Las pruebas de software (en inglés software testing) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o stakeholder. Es una actividad más en el proceso de control de calidad.

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo.

5.1.1. Tipos de pruebas de software

Pruebas de Compatibilidad

Se comprueba el funcionamiento del software desarrollado en muchas plataformas: sistemas operativos, navegadores, redes, hardware...entre otros

Pruebas de regresión

Se evalúa el correcto funcionamiento del software desarrollado frente a evoluciones o cambios funcionales.

Pruebas de Integración

Se centra principalmente en las comunicaciones y las conexiones entre los diferentes módulos del software desarrollado o con terceros (Publicidad, pasarelas de pago, etc.)

Pruebas funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático. Dicho de otro modo son pruebas específicas, concretas y exhaustivas para probar y validar que el software hace lo que debe y sobre todo, lo que se ha especificado. (Avella, Clara, Gómez, Juan, & Caro, Silvina, 2011)

Para este proyecto hemos definido dos pruebas funcionales:

5.1.2. Prueba funcional: Ingresar al sistema

Tabla 5.1: Test case: Login

Test Case: Login	
•	Curso Normal 1. El Usuario ejecuta el sistema mediante el acceso directo. 2. El Sistema presenta la **pantalla de Login**. 3. El Usuario ingresasus credenciales y presiona el botón **Ingresar**. 4. El Sistema valida las credenciales. 5. El Sistema presenta la pantalla principal.
•	Alternativo: Credenciales incorrectas 1. El Sistema presenta mensaje de error. 2. Vuelve al paso 3 del Curso Normal.

Autor: Alan Yamil Correa Requena

Este test case va a probar el ingreso de un usuario al sistema.

1. Ejecutar el sistema e introducir las credenciales

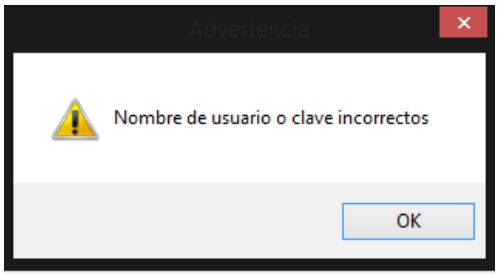
Figura 5.1: Pantalla de inicio del sistema

Fuente: El Sistema

Autor: Alan Yamil Correa Requena

2. Si se introducen credenciales incorrectas, el sistema presentará un mensaje

Figura 5.2: Mensaje del sistema



Fuente: El Sistema

Autor: Alan Yamil Correa Requena

3. Si las credenciales son correctas, el sistema mostrará la pantalla del formulario principal

Figura 5.3: Pantalla principal del sistema

Fuente: El Sistema

Autor: Alan Yamil Correa Requena

5.1.3. Prueba funcional: Insertar spot

Tabla 5.2: Case: Inseertar Spot

Test Case: Insertar Spot	
•	Curso Normal 6. El Usuario se encuentra en la **pantalla Spots** y presiona el botón **Nuevo**. 7. El Sistema habilita los campos necesarios. 8. El Usuario ingresa los datos. 9. El Usuario selecciona un archivo de video. 10. El Usuario presiona el botón **Guardar**. 11. El Sistema valida los datos. 12. El Sistema guarda en la base de datos. 13. El Sistema presenta mensaje. 14. El Sistema deshabilita los campos.
•	Alternativo: Error al guardar 3. El Sistema captura una excepción. 4. El Sistema presenta mensaje de error. 5. Vuelve al paso 5 del Curso Normal.

Fuente: El Sistema
Autor: Alan Yamil Correa Requena

Este test case comprobará la funcionalidad de insertar un spot en la base de datos.

Previamente se debe ingresar a la pantalla Spots.

1. Presionar el botón Nuevo

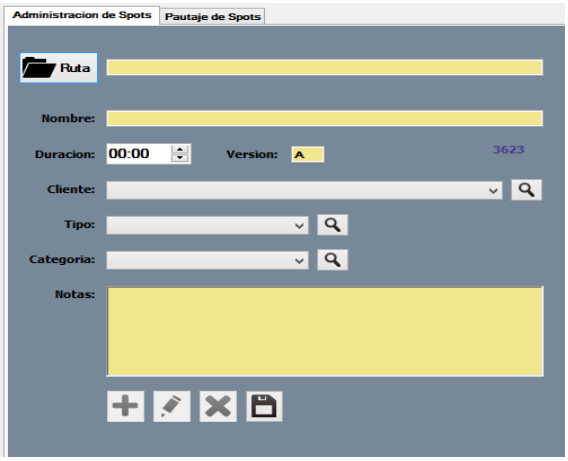
Figura 5.4: Pantalla spots – administración

Fuente: El Sistema

Autor: Alan Yamil Correa Requena

2. El sistema habilita los campos necesarios y los resalta con color amarillo

Figura 5.5: Pantalla spots – campos habilitados para inserción

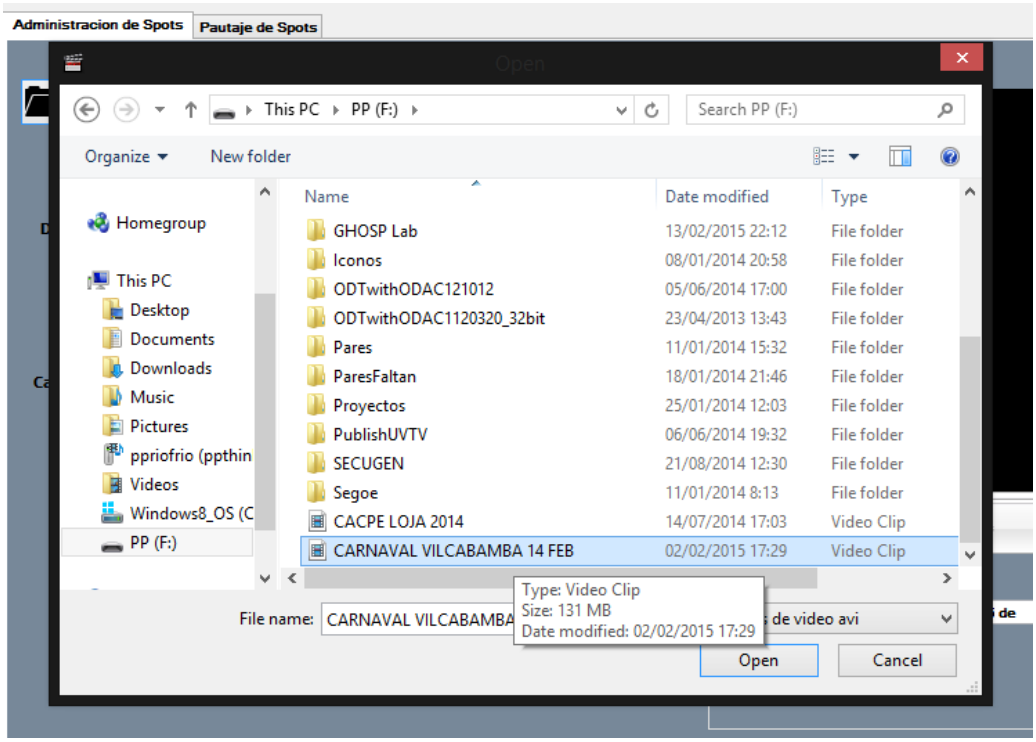


Fuente: El Sistema

Autor: Alan Yamil Correa Requena

El usuario debe ingresar todos los datos obligatorios. El principal es la ruta del spot. Para ello debe presionar el botón Ruta, el cual presentará el cuadro de diálogo Buscar Archivo de Video.

Figura 5.6: Abrir archivo de video

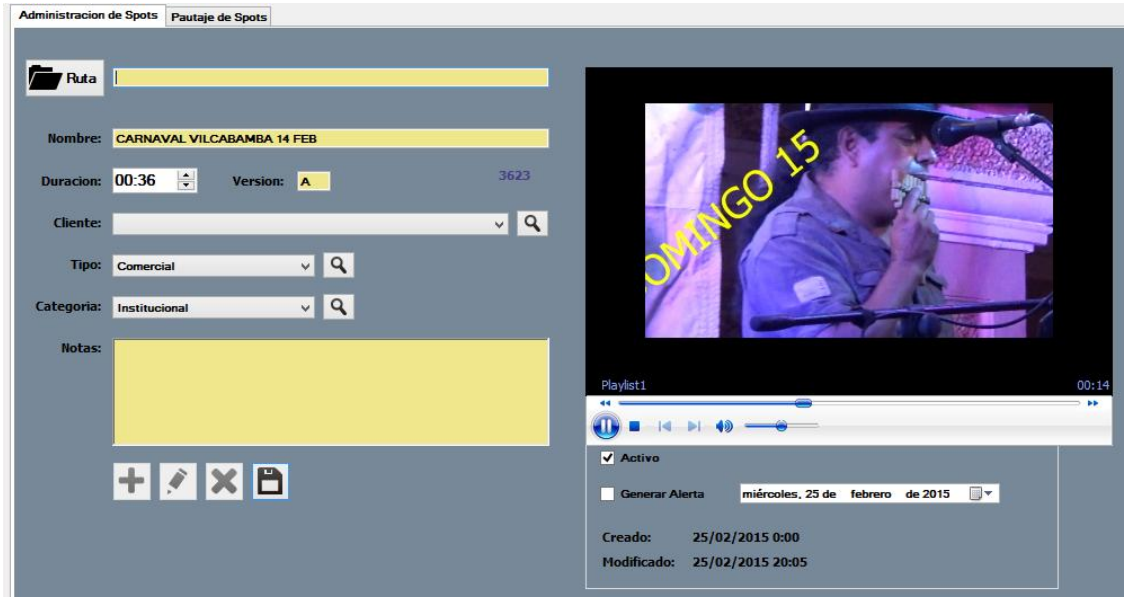


Fuente: El Sistema

Autor: Alan Yamil Correa Requena

El sistema empieza a reproducir el archivo de video elegido y toma los datos de nombre, ruta de archivo y duración.

Figura 5.7: Pantalla spots – reproduciendo archivo de video

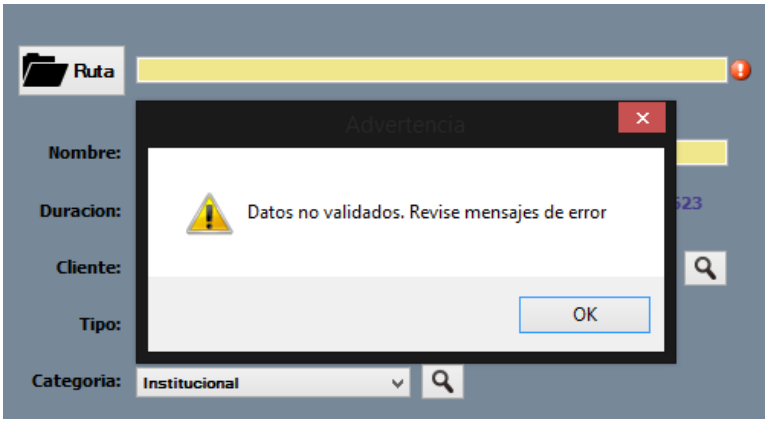


Fuente: El Sistema

Autor: Alan Yamil Correa Requena

3. El usuario presiona el botón guardar
4. El sistema valida los datos y si hay errores presenta mensajes junto a los controles que originan el error.

Figura 5.8: Mensaje de validación

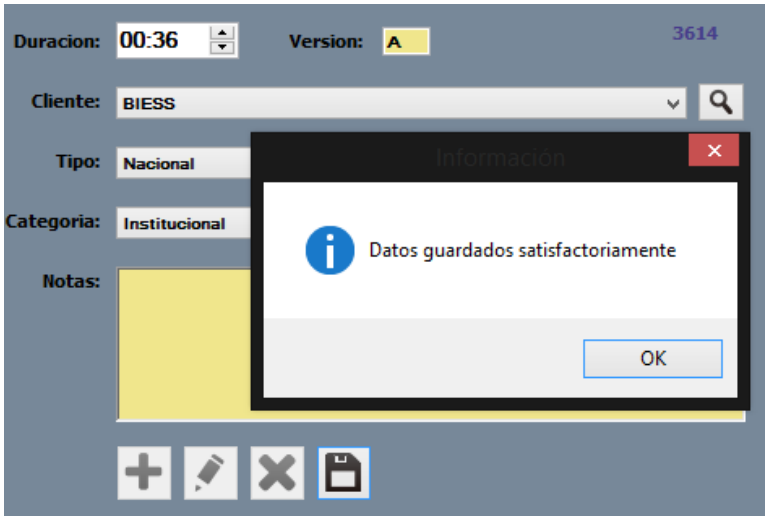


Fuente: El Sistema

Autor: Alan Yamil Correa Requena

Si los datos son correctos, el sistema guarda y presenta un mensaje informativo.

Figura 5.9: Mensaje de proceso completado con éxito



Fuente: El Sistema

Autor: Alan Yamil Correa Requena

6. Capítulo: Conclusiones, Recomendaciones, Bibliografía y Anexos

6.1. Conclusiones

El uso del Sistema de Control y Emisión de la publicidad en UV Televisión constituye una mejora global en los procesos. Analizando el método original y el propuesto en este proyecto, se tiene los siguientes resultados

- El registro de los spots y órdenes de emisión en una base de datos proporciona una forma de búsqueda eficiente y rápida con respecto a la búsqueda sobre los archivos tradicionales en carpetas físicas.
- Se aumenta un paso en el proceso, esto puede parecer contradictorio pero significa una ventaja muy grande sobre el proceso tradicional porque el paso de ingesta de información (registro de spot con sus horarios) reemplaza dos tareas muy engorrosas: la modificación de proyectos en Adobe Premiere y la emisión de certificados.
- El proceso de organización de la publicidad puede ser realizado en pocos minutos y los resultados se reflejan en tiempo real puesto que se puede acceder al sistema de forma remota a través de red de datos. Por ejemplo, si se necesita difundir un comercial nuevo, basta con registrarlo en el sistema y se emitirá en el siguiente corte comercial programado, si por el contrario es necesario dejar de difundir un spot en el acto, basta con cambiar su estado en el sistema y no será emitido. Frente al proceso tradicional que involucraba presencia física del organizador y demandaba espacios de tiempo libre, esto es una mejora indiscutible.

- La metodología ICONIX se ha ajustado perfectamente al proyecto porque su definición, en su mayoría gráfica, ayuda a detectar fallas en los procesos y a realizar un diseño óptimo. El método de prototipado permitió construir el software de acuerdo a los requerimientos de los usuarios y hacer una interfaz amigable para ellos.
- El desarrollo de controles de usuario y de formularios y controles personalizados, permiten definir poderosas herramientas de construcción de aplicaciones ya que se pueden abstraer atributos y métodos comunes, dejando a las instancias heredadas libres de código repetido para poder enfocar el desarrollo a la lógica de negocios.

6.2. Recomendaciones

- Adicionar la capacidad de automatizar bloques enteros de programación que incluyan tanto programas pre grabados como la publicidad que va dentro de los mismos.
- Adicionar librerías que permitan utilizar más tipos de tarjetas de video (este proyecto se limita al uso de tarjetas compatibles con DirectShow)
- Extender la funcionalidad del software con un módulo de edición de videos.
- Se podría enlazar a programas de facturación para emitir automáticamente facturas al finalizar la orden de emisión, basándose en el número de spots emitidos (cantidad) y en el horario emitido (costo).
- Definir políticas de programación y convenciones de nomenclatura. De preferencia investigar aquellos ya probados o reconocidos por la comunidad informática de forma periódica de modo que se utilice los más actuales.
- Manejar CrystalReports, si bien no es la herramienta predeterminada para elaborar reportes en Visual Studio, es una herramienta poderosa y fácil de utilizar que el generador de reportes que incluye el IDE. Por esto vale la pena realizar los pasos adicionales para utilizarla.
- Utilizar como tecnología de acceso a datos los DataSets con TableAdapters, ya que aceleran el proceso de desarrollo de la aplicación porque abstraen en gran parte la necesidad de digitar código, presentando una interfaz gráfica fácil de gestionar y mantener.
- La tecnología de vinculación de datos DataBinding de Visual Studio permite realizar con poco esfuerzo tareas de navegación y manipulación de datos en la

capa de interfaz de usuario que resultarían exhaustivas usando otros métodos. Por la sencillez de su uso, permiten que el código sea mucho más fácil de leer y mantener. Se recomienda su uso a todos los desarrolladores que usen Visual Studio.

6.3. Bibliografía

Avella, Clara, Gómez, Juan, & Caro, Silvina. (2011). *Aplicación de inspecciones y pruebas de software.* Universidad de Boyacá.

Bhphotovideo. (25 de 04 de 2006). Obtenido de http://www.bhphotovideo.com/c/browse/professional-video

Booch, Grady, Rumbaugh, James, & Jacobson, Ivar. (2005). *The Unified Modelling User Guide.Addison-Wesley Professional* (2ª ed.).

González, P. (2014). *Arquitectura y Diseño del Software.* Kindle Edition .

Larman, C. (2003). *UML y Patrones. 2da.* . Prentice Hall.

Microsoft. (s.f.). *Microsoft*. Obtenido de http://www.microsoft.com/en-us/server-cloud/products/sql-server/

Rosenberg, D. (2011). *ICONIX Process Roadmaps.* Fingerpress.

StevenPo. (07 de 07 de 2011). *msdn.microsoft*. Obtenido de http://msdn.microsoft.com/es-es/library/bz9tthwx(v=vs.80).aspx

StevenPo. (07 de 07 de 2011). *msdn.microsoft.* Obtenido de http://msdn.microsoft.com/es-es/library/bz9tthwx(v=vs.80).aspx

visualstudio. (12 de 11 de 2014). Obtenido de https://www.visualstudio.com/explore/application-development-vs

Wikipedia. (26 de 01 de 2015). *Wikipedia*. Obtenido de http://en.wikipedia.org/wiki/Software_Ideas_Modeler

6.4. Anexos

6.4.1. Anexo 1: Orden de emisión

PUBLICITAS PUBLICITAS CIA. ANON. DE PUBLICIDAD

Orden de Inserción de TV: 317368

U.V. TV. LOJA ASOCIADOS

ATENAS 0 Y PARIS

RUC: 1190050642001

Fecha de Emisión: 21/01/2014

Anunciante: MINISTERIO DE EDUCACION

Producto: INFORMATIVA

Campaña: INFORMATIVA / INFORMATIVA TV LOCAL ENE-FEB

Enero 2014

Ciudad	Espacio	Versión	Med	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	Ins	Consecio	FP	C. Unitario	Bonificación	Inversión
U.V. TELEVISION LOJA				X 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31						
LOJA	0600 - NOTICIEROS	INFORMATIVA- REFUERZO INSCRIP	30		6		☑	90.30		541.80
LOJA	1900 - NOTICIEROS	INFORMATIVA- REFUERZO INSCRIP	30		4		☑	90.30		361.20
Total U.V. TELEVISION LOJA				2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	10					903.00
Total Enero				2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	10					903.00
Total Documento					10					903.00

Bonificable	
Prefacturado	
Facturable	903.00
Cargo	
Total	903.00

Cliente/Agencia	Recibido	Totales	Ins	Inversión
		SubTotal Página 1	10	903.00
		Subtotal		903.00
		Total	**10**	**903.00**
		IVA (12%)		108.36
		Total Orden	**10**	**1,011.36**
Monica Cangas	(Ninguno)			

Facturar a:	Anunciante:	Observaciones
PUBLICITAS CIA. ANON. DE PUBLICIDAD fero. de Mayo 512 y Los Ríos.- Guayaquil. RUC: 0990094772001 Tel: 2283300	MINISTERIO DE EDUCACION AV. AMAZONAS N34-451 ENTRE ATAHUALPA Y JUAN PABLO SAENZ RUC: 1760001040001	Deberán facturar de acuerdo a indicaciones de la orden y deberán enviar una copia de la misma, requisito indispensable para el pago. La misma no podrá ser emitida hasta el 25 del mes, caso contrario la fecha de factura será el mes siguiente. Certificamos que las piezas publicitarias, concepto de transmisión de la presente orden, cumplen con todas las normas, reglamentos y leyes vigentes en el Ecuador.

Presupuesto: 163167

6.4.2. Anexo 2: Autorización al tesista.

6.4.3. Anexo 3: Manual de usuario

6.4.4. Anexo 4: Anteproyecto de tesis

6.4.5. Anexo 5: Reporte de emisión