CAPÍTULO II

RESUMEN

Al iniciar los lenguajes de programación era difícil hablar de una tecnología que abarque los problemas del desarrollo de software, el tiempo pasó y los lenguajes eran más robustos. Los investigadores llegaron a darle forma a los lenguajes de desarrollo organizando los procesos que estos contienen, sin embargo en la actualidad existen un sinumero de tecnologías orientadas al desarrollo de aplicaciones, esto para solucionar problemas a los usuarios quienes por salir a delante y ser mejores en la rama que se desempeñan desean fortalecer sus procesos y automatizarlos.

Por definición, es el personal técnico quien se preocupa más por la tecnología. El interés de muchos profesionales de software se centra, principalmente, en el modo de funcionamiento de las aplicaciones y no tanto en la forma de interacción que pueda darse entre éstas y los usuarios. Sin embargo, los usuarios dan gran importancia a las interfaces. La interfaz de una aplicación

constituye una parte fundamental de la experiencia global del usuario con el software particular. En lo que respecta a los usuarios, la aplicación es la experiencia. La experiencia mejorada de los usuarios mediante una interfaz optimizada puede contribuir al incremento de la productividad, a la generación de clientes leales y a una ampliación de las ventas en línea, entre muchas otras ventajas.

La tecnología que esta en auge en el siglo XXI da a conocer el potencial que tiene para con los programadores, esta tecnología lo que logra es unir el desarrollo y el diseño en una sola plataforma de desarrollo sin separar los procesos, es decir la persona que esta realizando una aplicación podrá realizar el diseño de su aplicación sin necesidad de que otras personas puedan influir en el sistema que el realiza. Otro area donde es visible el avance de las nuevas tecnologías es el manejo de datos (Base de Datos), antes la manipulación de datos de las grandes empresas solo se lo llevaban en archivos planos, y ahora existen varias opciones para almacenar la información de las empresas.

Con estas dos areas se puede decir que los lenguajes de programación y la manipulación de la información de una base de datos tratan de ir a la par dando soluciones más eficientes, este es el caso de .NET que incorpora la programación con el manejo de los datos desde diferentes bases de datos.

Pero si con Windows Forms también podemos incluir documentos, imágenes, gráficos 2D y 3D, video y audio usando Adobes's PDF, GDI+, Direct3D y Windows Media Player, respectivamente, entonces… ¿Por qué utilizar WPF?

Lo primero que hay que resaltar es que WPF no reemplaza a Windows Forms, de hecho esta última tecnología seguirá vigente, de tal manera que aún veremos nuevos desarrollos basados en Windows Forms. Igual de importante es remarcar que estas dos plataformas no son mutuamente excluyentes: WPF y Windows Forms pueden ser utilizadas en una misma aplicación ya que cada tecnología es capaz de albergar elementos de la interfaz de usuario definido por la otra.

Las dos plataformas tienen diferentes puntos fuertes y pueden complementarse mutuamente. Entre los puntos fuertes de Windows Forms se

puede resaltar el extensivo soporte para la conexión remota a bases de datos y el amplio rango de controles para la presentación de datos en tablas, para el manejo de fechas, horas y calendarios. Esto último es una de las ventajas de Windows Forms frente a WPF ya que WPF no cuenta con dichos controles teniendo que incorporarlos desde otras fuentes.

No obstante, la creación de interfaces de usuario modernas va más allá de la unificación de tecnologías diversas. También consiste en aprovechar las ventajas que ofrecen las tarjetas gráficas modernas. De este modo, WPF puede transferir la mayor carga de trabajo posible a cualquier unidad de procesamiento de gráficos (GPU) disponible en el sistema. Una interfaz moderna tampoco debe verse limitada por las deficiencias de los gráficos de mapa de bits. Por esta razón, WPF usa únicamente gráficos vectoriales, lo que permite que las imágenes se ajusten automáticamente al tamaño y a la resolución de la pantalla en la que se muestran. En lugar de crear gráficos diferentes para la presentación en monitores pequeños y en pantallas grandes, el desarrollador puede dejar que WPF se ocupe de adaptarlos.

Gracias a la unificación en una misma base de todas las tecnologías necesarias para crear interfaces de usuario, WPF puede simplificar enormemente la labor de quienes crean las interfaces. Sólo tendrán que familiarizarse con un único entorno, por lo que WPF puede reducir el costo asociado a la creación y el mantenimiento de aplicaciones. Además, al facilitar la generación de interfaces que incorporan gráficos y vídeo, entre otros elementos, WPF puede mejorar la calidad (y el valor comercial) de la interacción de los usuarios con las aplicaciones de Windows.

2. Estudio de la Tecnología WPF

2.1. Que es WPF?

WPF (Windows Presentation Foundation) o "Avalon", como inicialmente fue nombrado cuando se dio a conocer en el 2003, es una plataforma unificada e incluida en el Framework 3.0 que permite construir aplicaciones que combinan interfaces ricas de usuario con gráficos 2D y 3D, distintos tipos de documentos, contenidos multimedia, animaciones. WPF fue orientado a unificar los mecanismos de creación y gestión de interfaces de usuario.

Por definición, es el personal técnico quien se preocupa más por la tecnología. El interés de muchos profesionales de software se centra, principalmente, en el modo de funcionamiento de las aplicaciones y no tanto en la forma de interacción que pueda darse entre éstas y los usuarios. Sin embargo, los usuarios (que, al fin y al cabo, son los que pagan) dan gran importancia a las interfaces. La interfaz de una aplicación constituye una parte fundamental de la experiencia global del usuario con el software particular. En lo que respecta a los usuarios, la aplicación es la experiencia. La experiencia mejorada de los usuarios mediante una interfaz optimizada puede contribuir al incremento de la productividad, a la generación de clientes leales y a una ampliación de las ventas en línea, entre muchas otras ventajas.

Los usuarios, que un día se conformaban con interfaces basadas en caracteres, ahora se han acostumbrado a las interfaces gráficas. No obstante, los requisitos que deben cumplir estas interfaces siguen aumentando. El uso de gráficos y componentes multimedia se ha hecho más generalizado. Además, la Web ha condicionado a una generación de usuarios que esperan obtener una interacción con software sencilla. La relevancia de las interfaces de las aplicaciones crece con el aumento del tiempo y los usuarios dedican a interactuar con las mismas. Para satisfacer las crecientes expectativas, la tecnología de creación de interfaces de usuario debe avanzar a la par.

Windows Presentation Foundation (WPF) es un subsistema de presentación unificado para Windows, expuesto mediante WinFX, el modelo de programación de código administrado para Windows Vista que extiende

Microsoft .NET Framework. WPF se compone de un motor de visualización y un marco de código administrado. WPF unifica la forma en que Windows crea, muestra y manipula documentos, elementos multimedia e interfaces de usuario (UI), lo que permite a programadores y diseñadores crear experiencias de usuario diferenciadas y visualmente sorprendentes.

WPF se basa en el código administrado, pero usa un lenguaje de marcado, el lenguaje de marcado de aplicaciones extensible (XAML), para que la generación de aplicaciones sea mucho más fácil para los diseñadores. Las aplicaciones basadas en XAML son compatibles actualmente con C# y Microsoft Visual Basic .NET. Si escribe una aplicación WPF entera con código de procedimientos, puede usar cualquier lenguaje CLR (Common Language Runtime).

WPF es la parte de la plataforma que permite crear aplicaciones gráficas muy visuales que utilizan **el poder de aceleración 3D** de las tarjetas aceleradoras y los recursos de hardware de los mismo, como los puertos USB, Bluetooth, lectoras, cámaras, etc. Además es la evolución de lo que se conocía como Windows Forms, algo conocido para los que hacían aplicaciones *Win32* hace algunos años. **WPF** está optimizado para crear aplicaciones que corran en **Windows XP** y **Windows Vista**, y tiene la capacidad de crear aplicaciones standalone o que se conecten a fuentes en Internet, para descargar texto, imágenes, audio, video o cualquier otra cosa, e inclusive trabajar en modo desconectado. El funcionamiento de WPF está basado en el manejo de archivos XAML, lo que lo vuelve muy flexible a la hora de crear o actualizar el modo gráfico de una aplicación.

2.2. Beneficios del uso de la tecnología WPF

2.2.1. ¿Por qué usar WPF?

Pero si con Windows Forms también podemos incluir documentos, imágenes, gráficos 2D y 3D, video y audio usando Adobes's PDF, GDI+, Direct3D y Windows Media Player, respectivamente, entonces… ¿Por qué utilizar WPF?

Lo primero que hay que resaltar es que WPF no reemplaza a Windows Forms, de hecho esta última tecnología seguirá vigente, de tal manera que aún veremos nuevos desarrollos basados en Windows Forms. Igual de importante es remarcar que estas dos plataformas no son mutuamente excluyentes: WPF y Windows Forms pueden ser utilizadas en una misma aplicación ya que cada tecnología es capaz de albergar elementos de la interfaz de usuario definido por la otra.

Las dos plataformas tienen diferentes puntos fuertes y pueden complementarse mutuamente. Entre los puntos fuertes de Windows Forms se puede resaltar el extensivo soporte para la conexión remota a bases de datos y el amplio rango de controles para la presentación de datos en tablas, para el manejo de fechas, horas y calendarios. Esto último es una de las ventajas de Windows Forms frente a WPF ya que WPF no cuenta con dichos controles teniendo que incorporarlos desde otras fuentes.

Con respecto a WPF, entre sus puntos fuertes se puede resaltar lo que sería su principal objetivo: la capacidad de permitir a los desarrolladores construir aplicaciones ricas en contenido sin tener que incorporar otras tecnologías que en Windows Forms generalmente son difíciles de integrar; es decir, con WPF incluir documentos, animaciones, gráficos 2D y 3D, multimedia, es posible y más simple sin necesidad de agregar tecnologías tales como GDI+, Direct3D, Windows Media Player.

Otra capacidad que ofrece WPF es que permite a los diseñadores y desarrolladores trabajar de forma integrada en el proceso de desarrollo de aplicaciones, gracias a la introducción de un nuevo lenguaje llamado eXtensible Application Markup Language (XAML) que separa en archivos diferentes el diseño de la interfaz, de la implementación lógica. Gracias a esto, los diseñadores y los desarrolladores trabajan utilizando la misma herramienta. Así se soluciona el problema que se le presenta al desarrollador cuando debe implementar un diseño realizado con una herramienta de diseño que no tiene ninguna conexión con la herramienta que él utiliza para la implementación (resulta bastante complicado que un diseño realizado en Photoshop, por ejemplo, pueda trasladarse con el mismo aspecto a un formulario diseñado en Visual Studio). Esta división en dos archivos también

facilita los cambios en el diseño de la interfaz sin la necesidad de cambiar otras partes de la aplicación.

También hay que tener en cuenta que WPF se basa totalmente en gráficos vectoriales, superando la limitación de los gráficos de mapas de bits de los Windows Forms, lo que permite el ajuste automático de tamaño y de la resolución de la pantalla, en vez de crear diferentes gráficos, unos para pantallas pequeñas y otros para pantallas más grandes, además de la consiguiente reducción de peso en el tamaño de las dll's.

Concluyendo: quizás el cambio de Windows Forms a WPF no sea sencillo, ni de un día para el otro. Para conocer todas las nuevas funcionalidades que ofrece y el nuevo paradigma de programación basado en el lenguaje XAML, así como para saber aprovechar la tremenda potencia que la herramienta ofrece, se necesita de tiempo de estudio y mucha dedicación. Aún así, superando las dificultades mencionadas y, frente a las características y beneficios que WPF ofrece al hacer más fácil el desarrollo de aplicaciones, tanto para el diseñador y principalmente para el desarrollador, y la posibilidad de incorporar interfaces gráficas más modernas y visualmente más llamativas, son justificadas las respuestas de porque se debe utilizar la plataforma Windows Presentation Foundation en el desarrollo de nuevas aplicaciones.

2.3. Requerimientos de la Tecnología WPF

El objetivo de Windows Presentation Foundation (WPF) es proporcionar estos avances en el entorno de Windows. WPF se incluye en la versión 3.0 de Microsoft .NET Framework y permite crear interfaces que incorporan documentos, componentes multimedia, gráficos bidimensionales y tridimensionales, animaciones, características tipo web, etc. Al igual que el resto de los componentes de .NET Framework 3.0, WPF estará disponible para Windows Vista, Windows XP y Windows Server 2003, y su lanzamiento está programado para que se produzca al mismo tiempo que Windows Vista. Este artículo sirve de introducción a WPF y ofrece una descripción de sus distintos componentes. El objetivo es aclarar los problemas que trata esta tecnología y analizar las soluciones que proporciona WPF.

Los requerimientos necesarios para el correcto funcionamiento de la tecnología son los que se enumeran a continuación:

- Framework 3.0 o superior (En la actualidad se tiene la versión 4.0).
- Sistemas operatives Windows XP, Windows Vista, Windows 7, Windows Server 2003 o superior.
- Memoria RAM mínimo 1G.
- Procesador que tenga incluido acelerador gráfico y si no lo tiene deberá incluirlo para que las aplicaciones funcionen en su totalidad.

Editores de XAML

- XamlPadX.- Extensión del XamlPad que acompaña al SDK.
- Kaxaml.- Editor que acepta code snnipets y con una interfaz simple y agradable.
- XamlCruncher.- Editor de Charles Petzold que acompaña a su libro con el código fuente.

Herramientas de depuración y rendimiento

- Snoop.- Herramienta de Peter Blois' para depurar de forma visual aplicaciones WPF.
- Lutz Roeder's .NET Reflector.- Navegar entre el árbol de clases de un ensamblado.
- WPF Performance Tools.- Herramientas de rendimiento integrado como parte del SDK.
- UI Automation Stress.- pruebas de estrés para el entorno de nuestra aplicación WPF.

Herramientas para Interfaces en 3D

- ZAM 3D.- Utilidad para crear y animar objetos en 3D y exportarlos a XAML para usarlos en nuestras aplicaciones.
- 3D Tools for WPF.- librería que nos permite interactuar con elementos 2D en objetos 3D.

2.4. Ventajas de WPF

2.4.1. Plataforma unificada para interfaces de usuario modernas

Hasta la introducción de WPF, la creación de una interfaz de usuario de Windows como la descrita anteriormente requería el uso de varias tecnologías diferentes. En la siguiente tabla se presenta un resumen de la situación.

	Win Forms	PDF	Win Forms/ GDI+	Media Player	Direct3D	WPF
Interfaz gráfica, como formularios y controles	X	.	.	.	.	X
Documentos en pantalla	X	.	.	.	.	X
Documentos de formato fijo	.	X	.	.	.	X
Imágenes	.	.	.	X	.	X
Vídeo y audio	.	.	.	X	.	X
Gráficos bidimensionales	.	.	X	.	.	X
Gráficos tridimensionales	.	.	.	.	X	X

Tabla 2.1: Tabla de Interfaces Unificadas

Para crear formularios, controles y otros aspectos típicos de una interfaz gráfica de usuario de Windows, probablemente, un desarrollador seleccionará Windows Forms, uno de los componentes de .NET Framework. En caso de que la interfaz deba mostrar documentos, Windows Forms ofrece cierta compatibilidad con documentos en pantalla, mientras que los documentos de formato fijo suelen ser PDF de Adobe. En el tratamiento de imágenes y gráficos bidimensionales, el desarrollador usará GDI+, un modelo de programación definido al que también se puede obtener acceso mediante Windows Forms. Para reproducir vídeo y audio, elegirá Windows Media Player y, en lo que respecta a gráficos tridimensionales, usará Direct3D, un componente estándar de Windows.

Esta situación tan compleja se debe exclusivamente a motivos históricos y no tiene mucho sentido. Lo lógico es contar con una solución única consolidada: WPF. Al crear aplicaciones para equipos que tengan instalada la aplicación WPF, lo más probable es que el desarrollador use esta tecnología para cubrir todos los aspectos mencionados. Después de todo, ¿por qué no usar una base coherente para la creación de interfaces de usuario en lugar de un conjunto variado de tecnologías independientes?

WPF no reemplaza todo lo incluido en esta lista, por supuesto. Las aplicaciones de Windows Forms conservarán su valor e, incluso en un entorno de WPF, algunas aplicaciones nuevas continuarán usando Windows Forms. Es importante recordar que WPF puede interoperar con Windows Forms. Windows Media Player mantendrá su función independiente y se seguirán usando documentos PDF. Direct3D no perderá su gran valor tecnológico en juegos y otros tipos de aplicaciones. De hecho, WPF hace uso de Direct3D para todas las tareas de representación.

Sin embargo, al proporcionar una amplia gama de funciones en una sola tecnología, WPF simplifica de forma significativa la creación de interfaces de usuario modernas. Como ejemplo de lo que ofrece este enfoque unificado.

No obstante, la creación de interfaces de usuario modernas va más allá de la unificación de tecnologías diversas. También consiste en aprovechar las ventajas que ofrecen las tarjetas gráficas modernas. De este modo, WPF puede transferir la mayor carga de trabajo posible a cualquier unidad de procesamiento de gráficos (GPU) disponible en el sistema. Una interfaz moderna tampoco debe verse limitada por las deficiencias de los gráficos de mapa de bits. Por esta razón, WPF usa únicamente gráficos vectoriales, lo que permite que las imágenes se ajusten automáticamente al tamaño y a la resolución de la pantalla en la que se muestran. En lugar de crear gráficos diferentes para la presentación en monitores pequeños y en pantallas grandes, el desarrollador puede dejar que WPF se ocupe de adaptarlos.

Gracias a la unificación en una misma base de todas las tecnologías necesarias para crear interfaces de usuario, WPF puede simplificar enormemente la labor de quienes crean las interfaces. Sólo tendrán que familiarizarse con un único entorno, por lo que WPF puede reducir el costo asociado a la creación y el mantenimiento de aplicaciones. Además, al facilitar la generación de interfaces que incorporan gráficos y vídeo, entre otros elementos, WPF puede mejorar la calidad (y el valor comercial) de la interacción de los usuarios con las aplicaciones de Windows.

2.4.2. Posibilidad para desarrolladores y diseñadores de trabajar conjuntamente

Es estupendo disponer de una base tecnológica unificada para crear interfaces de usuario completas. Pero quizás sea pedir demasiado que los desarrolladores sepan sacar el máximo partido de este entorno para crear interfaces completas y fáciles de usar. La creación de interfaces de usuario eficaces, en especial cuando se trata de interfaces globales requiere a menudo conocimientos que muchos profesionales de software no tienen. Aunque resulta posible crear aplicaciones sin recurrir a esta experiencia, lo cierto es que para crear una gran interfaz de usuario, es necesario trabajar con diseñadores de interfaces profesionales.

Pero, ¿cómo conseguir que diseñadores y desarrolladores trabajen en equipo? La forma actual de interacción de estas dos disciplinas es problemática. Por lo general, el diseñador usa una herramienta gráfica para crear imágenes estáticas de los diseños de pantalla que debe mostrar una aplicación específica. A continuación, las imágenes pasan al desarrollador, quien se encarga de crear el código necesario para convertirlas en realidad. Sin embargo, para un desarrollador, puede ser muy complicado o incluso imposible implementar las imágenes que el diseñador dibuja tan fácilmente. Las limitaciones tecnológicas, las programaciones apretadas, la falta de conocimientos, los malentendidos o simples desacuerdos pueden hacer que el desarrollador no refleje completamente la visión del diseñador. Se necesita un método de trabajo mejorado que permita la colaboración estrecha del personal de estas disciplinas interdependientes, sin que esto influya de forma negativa en la calidad de la interfaz.

Con este fin, WPF incluye el lenguaje de marcado de aplicaciones extensible (XAML). El lenguaje XAML define elementos XML, como Button, TextBox, Label, entre muchos otros, para especificar exactamente la apariencia de las interfaces de usuario. Asimismo, los elementos XAML suelen disponer de atributos, lo que permite definir varias opciones. Por ejemplo, este sencillo fragmento de XAML sirve para crear un botón rojo que contiene la palabra "No":

```
<Button Background="Red">
```

```
No
</Button>
```

Cada elemento XAML corresponde a una clase de WPF. A su vez, cada atributo de dicho elemento cuenta con una propiedad o evento correspondiente en la clase. Por ejemplo, es posible crear el mismo botón rojo con el código C# siguiente:

```
Button btn = new Button();
btn.Background = Brushes.Red;
btn.Content = "No";
```

Pero, si todo lo que puede expresarse en XAML, puede expresarse también en código, ¿para qué sirve XAML? Lo cierto es que resulta mucho más fácil crear herramientas que generan y usan descripciones basadas en XML, que usar código con la misma finalidad. XAML ofrece un método basado en herramientas muy sencillo para describir interfaces de usuario y, de este modo, permite una mejor colaboración entre desarrolladores y diseñadores. En la siguiente ilustración se muestra el proceso.

Mediante el uso de una herramienta como Microsoft Expression Interactive Designer, el diseñador puede definir la apariencia y el modo de interacción de una interfaz de usuario. La herramienta, diseñada específicamente para la definición de la apariencia de interfaces de WPF, genera una descripción de la interfaz expresada en XAML. Aunque podría incluir un botón tan sencillo como el que se muestra en el ejemplo, en realidad, la descripción es mucho más compleja de lo que sugiere el fragmento de código anterior.

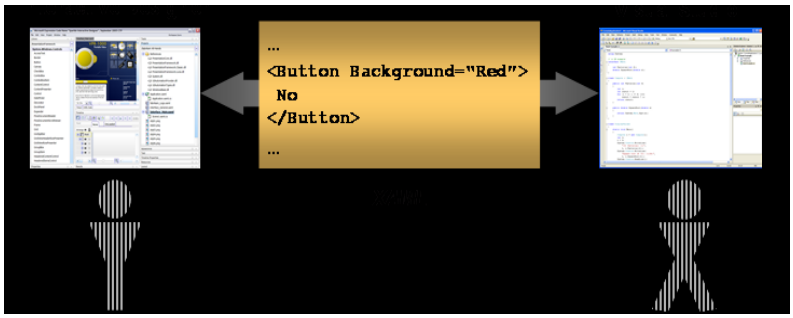

Figura 2.1: XAML Permite a Desarrolladores y diseñadores trabajar juntos

Seguidamente, el desarrollador importa la descripción XAML en una herramienta como Microsoft Visual Studio. En lugar de volver a crear por completo la interfaz a partir de las imágenes estáticas generadas por el diseñador, la definición de la interfaz se adopta de forma sistemática. A continuación, el desarrollador escribe el código de la interfaz, como los controladores de eventos, junto con el resto de las funciones que requiera la aplicación. También es posible crear estilos aplicables globalmente a la interfaz de aplicación, lo que facilita su personalización para situaciones diferentes.

El trabajo conjunto de diseñadores y desarrolladores reduce los errores de traducción que suelen darse cuando los desarrolladores implementan interfaces a partir de las imágenes creadas por los diseñadores. Además, permite el trabajo en paralelo del personal de estas dos disciplinas. Como resultado, se obtienen iteraciones más rápidas y comentarios más eficaces. Otra ventaja es que ambos entornos usan el mismo sistema de creación, por lo que las aplicaciones de WPF pueden pasar de un entorno de desarrollo a otro sin complicaciones. Existen herramientas más especializadas para el diseño de interfaces definidas mediante XAML como, por ejemplo, ZAM 3D de Electric Rain, que permite crear elementos de interfaz tridimensionales.

Las interfaces de usuario mejoradas pueden incrementar la productividad; ofrecen un valor comercial apreciable. No obstante, la creación de interfaces verdaderamente efectivas, sobre todo en un entorno tan polifacético como el que presenta WPF, requiere la participación total y en primera instancia de los diseñadores. Uno de los objetivos principales de XAML y de las herramientas compatibles es convertir esta posibilidad en realidad.

2.4.3. Tecnología común para interfaces de usuario de Windows y explorador web

Es importante crear interfaces de usuario eficaces para las aplicaciones de Windows. Y la creación de interfaces eficaces para aplicaciones basadas en Web es, por lo menos, igual de relevante. Por definición, estas interfaces vienen proporcionadas por el explorador web y lo más sencillo es dejar que el explorador muestre de forma pasiva el código HTML que recibe. Las interfaces

de explorador de mayor respuesta ofrecen ejecución lógica en JavaScript, para lo que pueden hacer uso de Java Script y XML (AJAX) de forma asincrónica. Estas interfaces pueden incluso admitir animaciones, vídeo y muchos otros componentes mediante Flash Player de Adobe y otras tecnologías. Este tipo de software web, conocido a veces como *aplicaciones de Internet enriquecidas*, puede mejorar considerablemente la experiencia del usuario, ya que pone a su alcance interfaces de funcionalidad total. Además, puede incrementar el valor comercial al dotar de un mayor atractivo a las aplicaciones web.

Hasta ahora, la generación de este tipo de interfaz requería el uso de un conjunto de tecnologías totalmente distintas a las usadas en la creación de interfaces nativas de Windows. Por ello, los desarrolladores suelen centrarse en uno de los siguientes enfoques: si se trata de un desarrollador de interfaces de Windows o de un desarrollador de interfaces web. Sin embargo, ¿por qué debe existir esta dicotomía para las aplicaciones de Internet enriquecidas con acceso a través de Windows? No hay motivo inherente que impida el uso de las mismas tecnologías tanto para interfaces nativas de Windows como para interfaces de explorador web.

WPF ofrece esta posibilidad. Así, un desarrollador puede crear una aplicación XAML del explorador (XBAP) con WPF, que se ejecuta en Internet Explorer. De hecho, es posible usar el mismo código para crear una aplicación de WPF independiente y una XBAP. Como ejemplo, la pantalla a continuación muestra una aplicación de servicios financieros que se ejecuta como aplicación de Windows independiente. Al igual que la aplicación hospitalaria anterior, esta combina texto, imágenes y varios tipos de gráficos. Además, la pantalla muestra el escritorio de Windows Vista, con elementos como el reloj y el tema Aero, que proporciona los bordes semitransparentes de la ventana de la aplicación.

Ahora, en lugar de ejecutarse en su propia ventana, la interfaz se presenta enmarcada por el explorador; sin embargo, su funcionalidad es exactamente la misma. También se puede usar el mismo código en ambos casos, lo que reduce el esfuerzo que requiere el tratamiento de los dos tipos de interfaces. Y, por supuesto, el uso del mismo código implica el uso de unos mismos conocimientos de desarrollo. Los desarrolladores ya no se verán obligados a

elegir entre la creación de interfaces de Windows y la generación de interfaces web. WPF elimina por completo esta distinción y permite que los mismos conocimientos se usen en ambos casos.

Otra ventaja derivada del uso de la misma tecnología para interfaces de Windows y web es que el creador de una aplicación no necesitará decidir de antemano el tipo de interfaz que debe presentar la aplicación. Siempre y cuando los clientes de destino sean compatibles con la ejecución de XBAP, la aplicación podrá ofrecer una interfaz de Windows o web (o ambas) con el mismo código en gran medida.

La descarga de XBAP se lleva a cabo a petición desde un servidor web, por lo que los requisitos de seguridad asociados son más estrictos que en el caso de aplicaciones de Windows independientes. Por consiguiente, las XBAP se ejecutan en un recinto de seguridad proporcionado por la seguridad de acceso a código de .NET Framework. Adicionalmente, las XBAP sólo se ejecutan en Windows con la tecnología WPF instalada en el sistema, y únicamente en las versiones de Internet Explorer 6 y 7. No obstante, en lo que respecta a las aplicaciones que cumplen estos requisitos, las aplicaciones de Internet enriquecidas pueden usar ahora la misma base que usan las aplicaciones de Windows independientes.

2.5. Herramientas para WPF

La familia de herramientas de *Microsoft Expresion* ha sido una salvación para la implementación de aplicaciones WPF, al comenzar esta tecnología era muy difícil diseñar sistemas ya que no existían editores potentes, solo tocaba conformase con el editor de Visual .NET la cual no ayudaba mucho, era como cuando se comenzó a programar JAVA, cuando tocaba realizar en block de notas, pero en la actualidad las herramientas de Microsoft Expresion han logrado que las aplicaciones se las desarrollen de forma rápida.

Microsoft Expresion consta de las siguientes herramientas cada una apta para cada necesidad a la hora de desarrollar sistemas WPF.

Expression Web.- Que está disponible y esta nos enfocada al cumplimiento de estándares, ya que permite validar nuestro contenido contra el estándar

que el usuario desee seguir o navegadores en los que se planea montar la aplicación Web. Además, Expression Web es compatible con hojas de estilo CSS y con Microsoft Visual Studio. Asimismo, incluye la capacidad de procesar archivos XML mediante Java Script.

Desde la versión 2 del programa, se permite la integración con lenguajes de servidor como ASP.NET o PHP sin necesitar de instalar un servidor. También permite la interacción con Adobe Photoshop para generar imágenes.

Expression Design.- De igual forma está disponible, (antes llamado Expression Graphic Designer, cuyo nombre en código era Acrylic) es una herramienta desarrollada por Microsoft, para los profesionales creativos y desarrolladores que quieran crear gráficos para su aplicación en interfaces de usuario, la web, o cualquier otro medio, ofreciendo potentes funciones que ayudarán a expresar la creatividad.

Expression Design, es una herramienta poderosa tanto para el desarrollador como para el diseñador, que permite crear increíbles gráficos vectoriales usando herramientas sencillas de diseño, permitiendo trabajar sobre gráficos y artes sin ningún problema, agregando efectos y elaborando capas; una herramienta que además de poderosa se destaca por su sencillez.

Expression Design 2.- Es el compañero perfecto para Expression Blend 2 aprovecha su obra de arte existente, o el uso intuitivo y flexible de herramientas de dibujo para crear rápidamente sofisticadas vector atractivos, a través de la transferencia de archivos a su xaml con Expression Blend 2 desarrollará proyectos que todo diseñador o programador ha deseado.

Expression Design.- (Antes llamado Expression Interactive Designer, cuyo nombre en código era Sparkle) es una herramienta profesional desarrollado por Microsoft, de diseño que le permite controlar la eficacia del XAML, .NET y Silverlight para proporcionar experiencias de usuario atractivas en escritorios conectados y Web.

es la herramienta de diseño profesional para crear experiencias de usuario atractivas y conectadas a la Web para Windows, abriendo la puerta a la

creación de diseños de interfaces de usuario mediante una amplia gama de tipos de medios, trabajando en un entorno de diseño en tiempo real.

Expression Blend 2.- Es su herramienta para aprovechar la potencia de XAML, .NET Framework 3,5 y Silverlight para obligar a entregar la mejor experiencia al usuario de ordenadores personales conectados y la red.

Novedades de Expression Blend

Expression Blend 2, en esta versión, hay mejoras significativas en la interoperabilidad con XAML, y Silverlight.

- Mejoras con Silverlight, Expression Blend 2 soporta proyectos Silverlight 1.0, que son creados como proyectos de sitio web en lugar de proyectos de aplicación, esto ayuda a mejora la interoperabilidad con Visual Studio 2008, pudiendo modificar un proyecto Silverlight 1.0 en cualquier momento, Expression Blend 2 incluye un editor de texto de Java Script para editar los códigos Silverlight 1.0 detrás de los archivos.

- Cambios en los archivos sincronizados, Al modificar un proyecto Silverlight en Expression Blend 2 y Visual Studio al mismo tiempo, cualquier cambio que se haga en Visual Studio se aplicará inmediatamente en Expression Blend 2.

- Storyboard Picker, sustituye a la antigua caja de Storyboard, picker se compone de una etiqueta que indique el nombre del storyboard seleccionado (un guión, si se selecciona), pudiendo acceder a éste a través del menú de acceso directo (disponible cuando usted haga clic derecho en la etiqueta), un botón de atajo (y el consiguiente menú contextual), un botón cerrar, para cerrar todos los storyboards además de la salida modo de grabación.

- Clip camino de edición y animación, Expression Blend 2 ahora permite modificar la saturación en los artboard después de crearlos, pudiendo animar cada uno de los vértices de saturación.

Arquitectura de extensibilidad de WPF Designer

Windows Presentation Foundation (WPF) Designer for Visual Studio es un entorno de edición visual para controles compuestos de WPF, implementados por el tipo *UserControl*.

WPF Designer se basa en un marco de trabajo con una arquitectura extensible, que puede ampliar para crear su propia experiencia de diseño personalizada.

Extendiendo el modelo de objetos de WPF Designer, puede personalizar en gran medida el aspecto y el comportamiento del contenido de WPF en tiempo de diseño. Por ejemplo, puede extender WPF Designer de las maneras siguientes:

- Personalizar los glifos de movimiento y cambio de tamaño con gráficos mejorados.
- Agregar un glifo a la superficie de diseño que incline el control seleccionado al mover el mouse.
- Modificar el aspecto y el comportamiento de un control en tiempo de diseño en las distintas herramientas.

La arquitectura de WPF Designer admite toda la fuerza expresiva de WPF. Esto permite crear numerosas experiencias de diseño visual que antes no eran posibles.

Modelo de objetos de WPF Designer

El modelo de objetos de WPF Designer es modular, lo que significa que al extender el tiempo de diseño, únicamente se extienden los elementos necesarios para sus propias características.

No tiene que escribir gran cantidad de código de compatibilidad para habilitar sus características de diseño personalizadas.

El modelo de objetos está compuesto de cinco unidades funcionales, que se describen en la tabla siguiente.

Unidad funcional	Descripción
Modelo de edición	Interfaz de programación para los objetos del diseñador.
Proveedores de características	Punto de extensibilidad principal en el marco de trabajo del diseñador.
Contexto de edición	Almacén central para el estado de un diseñador.
Herramientas	Herramientas para procesar los datos proporcionados por el usuario.
Almacén de metadatos	Almacén que contiene el comportamiento de un control en tiempo de diseño, para separar físicamente la lógica del diseñador de la lógica de tiempo de ejecución.

Tabla 2.2: Modelo de Objetos de WPF Designer

En la ilustración siguiente se muestra el modelo de objetos de WPF Designer.

Modelo de edición

El entorno de diseño interactúa con los controles de tiempo de ejecución mediante una interfaz de programación denominada modelo de edición. El modelo de edición está compuesto de tres subunidades funcionales: un modelo, un contenedor público que abstrae el modelo y una vista que representa la interfaz de usuario (IU) del modelo.

El entorno de diseño utiliza el tipo *ModelItem* para comunicarse con el modelo subyacente. Todos los cambios se realizan en los contenedores *ModelItem*, que afectan al modelo subyacente. Esto permite que el modelo sea simple. Los contenedores *ModelItem* administran las características complejas del diseñador, tales como la compatibilidad de transacciones, la traza de acciones deshechas y las notificaciones de cambios.

La clase *ModelService* proporciona el punto de entrada para el modelo de edición y para las notificaciones de eventos globales.

La clase *ViewService* asigna las representaciones visuales a los elementos del modelo subyacente.

Ambos servicios son necesarios para que el diseñador funcione. La clase *DesignerView*, que es responsable de procesar los datos proporcionados por el usuario y enrutarlos a los comandos, necesita estos dos servicios para asignar con precisión los datos proporcionados por el usuario al modelo.

Contexto de edición

En un diseñador en ejecución se acumula una cantidad significativa de información de estado. Por ejemplo, el estado del diseñador puede incluir qué objetos están seleccionados o el comportamiento que se produce cuando se presiona el botón primario. El estado del diseñador se almacena en una ubicación central para poder encontrarlo cuando se necesita. La clase *Editing Context* representa este repositorio central de estados para el diseñador.

La clase *Editing Context* separa el estado en dos categorías: datos y comportamiento. Los datos se almacenan como una tabla de elementos de contexto y el comportamiento se almacena como una tabla de servicios. Ambas tablas se indizan mediante una clave basada en tipos y son numerables.

La clase *Context Item* contiene un único fragmento de estado en el diseñador. Los elementos de contexto son inmutables, pero puede haber elementos de contexto nuevos que reemplacen a los existentes para simular la mutabilidad.

Se tiene acceso a los servicios a través de una propiedad Services, que devuelve una instancia de *ServiceManager*, y se tiene acceso a los elementos de contexto a través de una propiedad Items, que devuelve una instancia de *ContextItemManager*.

Comandos, tareas y herramientas

La arquitectura de herramientas de WPF Designer está compuesta de comandos, tareas y herramientas.

Un comando es un identificador único que representa un determinado comportamiento. Por ejemplo, "Cut" es un comando que significa que se debe cortar el texto seleccionado y agregarlo al Portapapeles. El código que implementa "Cut" varía según las distintas aplicaciones e incluso dentro de una misma aplicación. Por ejemplo, cortar texto en un documento de Word constituye una implementación diferente que cortarlo en el cuadro de texto de búsqueda del mismo documento. Independientemente de la implementación, el comando "Cut" permanece constante.

WPF Designer aumenta el sistema de comandos de WPF introduciendo un concepto de comando de herramienta. Un comando de herramienta implementa la interfaz *ICommand* y es como la clase *RoutedCommand*.

Una tarea tiene una colección de enlaces de comando que permite agregar los comandos enrutados. La clase *DesignerView* tiene código que utiliza la misma estrategia del enrutamiento que los comandos de herramienta para buscar y ejecutar comandos enrutados que se definen en tareas. La clase *DesignerView* habilita tareas que admiten comandos de WPF comunes, como *Copy*.

Una herramienta es una clase que procesa los datos proporcionados por el usuario. Todos los datos proporcionados por el usuario entran en el diseñador como uno o más eventos de entrada. Esos eventos de entrada se pasan a la herramienta actualmente activa, que los convierte en enlaces de entrada. Si se devuelve un enlace de entrada, se ejecuta el comando dentro del enlace.

Una herramienta puede representar el modo global del diseñador. Por ejemplo, si el usuario está seleccionando componentes en la superficie de diseño, ese modo de selección es posible porque la herramienta activa proporciona enlaces y comandos de entrada que administran la selección. Cuando el usuario crea una nueva instancia de un control, se activa una herramienta diferente que proporciona un conjunto diferente de comandos, que se enlazan a los mismos enlaces de entrada.

Almacén de metadatos

En el marco de trabajo de WPF Designer, los metadatos que definen el comportamiento de un control en tiempo de diseño se factorizan en un ensamblado independiente que se denomina almacén de metadatos. En .NET Framework 3.5, el almacén de metadatos se implementa en tablas de atributos basadas en código, con un archivo XML externo que hace referencia al código del diseñador y un perfil.

Las distintas herramientas pueden proporcionar diversos almacenes de metadatos con implementaciones totalmente diferentes del diseñador. Esto desacopla el comportamiento en tiempo de ejecución y en tiempo de diseño,

para que pueda revisar el diseñador de manera independiente con respecto al control.

2.6. Modelo de aplicación WPF

Según lo explicado anterior mente el desarrollo de sistemas con WPF, la parte de desarrollo y diseño van a la par, en un mismo entorno de programación ya que si se desarrolla un tema para un control, este nos servirá para todos los controles del mismo tipo y así evitar el diseño varias veces como sucede en Windows forms.

Para explicar mejor este punto realizaremos un diagrama explicativo del modelo más usado, ya que esto queda abierto a la expectativa del diseñador.

WPF aplicaciones con el patrón de diseño Model-View-ViewModel

El desarrollo de la interfaz de usuario de una aplicación de software profesional no es fácil. Puede ser una mezcla turbia de datos, interacción con el diseño, diseño visual, conectividad, subprocesamiento múltiple, seguridad, internacionalización, validación, las pruebas unitarias y un toque de vudú. Tener en cuenta que una interfaz de usuario expone el sistema subyacente y debe satisfacer los requisitos de estilo imprevisibles de sus usuarios, puede ser el área más volátil de muchas aplicaciones.

Los patrones son más complicadas, más probable que sea accesos directos utiliza más adelante que comprometer todos los esfuerzos anteriores para realizar acciones en la forma correcta.
No siempre es la culpa de los patrones de diseño. En ocasiones, utilizamos patrones de diseño complicado, que requieren escribir una gran cantidad de código porque la plataforma de interfaz de usuario en uso no admite propio bien a un modelo más sencillo. Qué es necesario es una plataforma que facilita la generación de interfaces de usuario con los patrones de diseño sencilla, time-tested (pruebas a tiempo) aprobado el desarrollador. Afortunadamente, Windows Presentation Foundation (WPF) ofrece exactamente.

A medida que el mundo de software continúa para que adopten WPF a un ritmo creciente, la comunidad WPF ha desarrollando su propia ecosistema de

patrones y prácticas. Aprovechando algunas características principales de WPF en combinación con el modelo de diseño Model-View-ViewModel (MVVM), guiará a través de un programa de ejemplo que muestra lo sencillo puede ser crear una aplicación WPF "forma correcta".

Al final de este artículo, será cómo borrar las plantillas de datos, comandos, el enlace de datos, el sistema de recursos y la trama MVVM todos los encajan para crear un marco sencillo, comprobable, eficaz en qué WPF cualquier aplicación puede prosperan. El programa de demostración que acompaña a este artículo puede servir como una plantilla para una aplicación WPF real que utiliza MVVM como su arquitectura principal.

Orden frente a caos

Es innecesario y contraproducente utilizar patrones de diseño para un sencillo programa "Hola mundo". Cualquier desarrollador competente puede entender unas pocas líneas de código de un vistazo. Sin embargo, a medida que aumenta el número de características en un programa, el número de líneas de código y partes móviles aumenta en consecuencia. Finalmente, la complejidad de un sistema y los problemas recurrentes contiene, anima a los desarrolladores organizar su código de tal forma que es más fácil comprender, explique, extender y solucionar problemas. Se disminuye el caos cognitivas de un sistema complejo al aplicar nombres conocidos a determinadas entidades en el código fuente. Se determinar el nombre para aplicar a un fragmento de código por considerar su función en el sistema.

Los desarrolladores a menudo intencionadamente estructuran su código según un modelo de diseño, en contraposición a permitir que los patrones surjan organizada mente. No hay nada malo en cualquier enfoque, pero en este artículo, examinan las ventajas de utilizar explícitamente MVVM como la arquitectura de una aplicación WPF. Los nombres de ciertas clases incluyen condiciones conocidas desde el modelo MVVM, tales como terminen con "ViewModel" si la clase es una abstracción de una vista. Este enfoque ayuda a evitar el caos cognitivas que se ha mencionado anteriormente. En su lugar,

por suerte puede existir en un estado de caos controlado, que es el estado natural de asuntos de proyectos de desarrollo de software más profesionales.

La evolución de Model-View-ViewModel

Empezaron desde personas crear interfaces de usuario de software, ha habido patrones de diseño populares para facilitar. Por ejemplo, el patrón Model-View-Presenter (MVP) ha disfrutado popularidad en varias plataformas de programación de interfaz de usuario. MVP es una variación del diseño Model-View-Controller, que ha sido el mejor alrededor de varias décadas. En caso de que nunca ha usado la trama de MVP antes, aquí es una explicación simplificada.

Lo que ve en la pantalla es la vista, los datos que se muestra están el modelo y el moderador enlaza los dos juntos. La vista se basa en un moderador para rellenarla con los datos de modelo, reaccionar a datos proporcionados por el usuario, proporcionar validación (quizá mediante la delegación en el modelo) y otras tareas.

En 2004, Martin Fowler publicado un artículo acerca de un patrón denominado Modelo de presentación (CP). El modelo de administrador de proyecto es similar a MVP en que separa una vista de su comportamiento y el estado. La parte interesante de la trama de administrador de proyecto es que se crea una abstracción de una vista, denominado el modelo de presentación. Una vista, a continuación, se convierte en simplemente una representación de un modelo de presentación. En la explicación del Fowler, muestra que el modelo de presentación actualiza con frecuencia la vista, para que los dos permanezcan sincronizados entre sí. Esa lógica de sincronización existe como código de las clases de modelo de presentación.

En 2005, Unveiled Gossman de John, actualmente uno de los arquitectos de WPF y Silverlight en Microsoft. MVVM es idéntico al modelo de presentación del Fowler, que ambos modelos característica una abstracción de una vista, que contiene el estado y el comportamiento de una vista.

Fowler introdujo el modelo de presentación como un medio de creación de una abstracción de interfaz de usuario independiente de la plataforma de una vista, mientras que Gossman introdujo MVVM como una forma estándar para aprovechar las características principales de WPF para simplificar la creación de interfaces de usuario. En ese sentido, considere MVVM que una especialización de la trama más general de administrador de proyecto, tailor-made para las plataformas WPF y Silverlight.

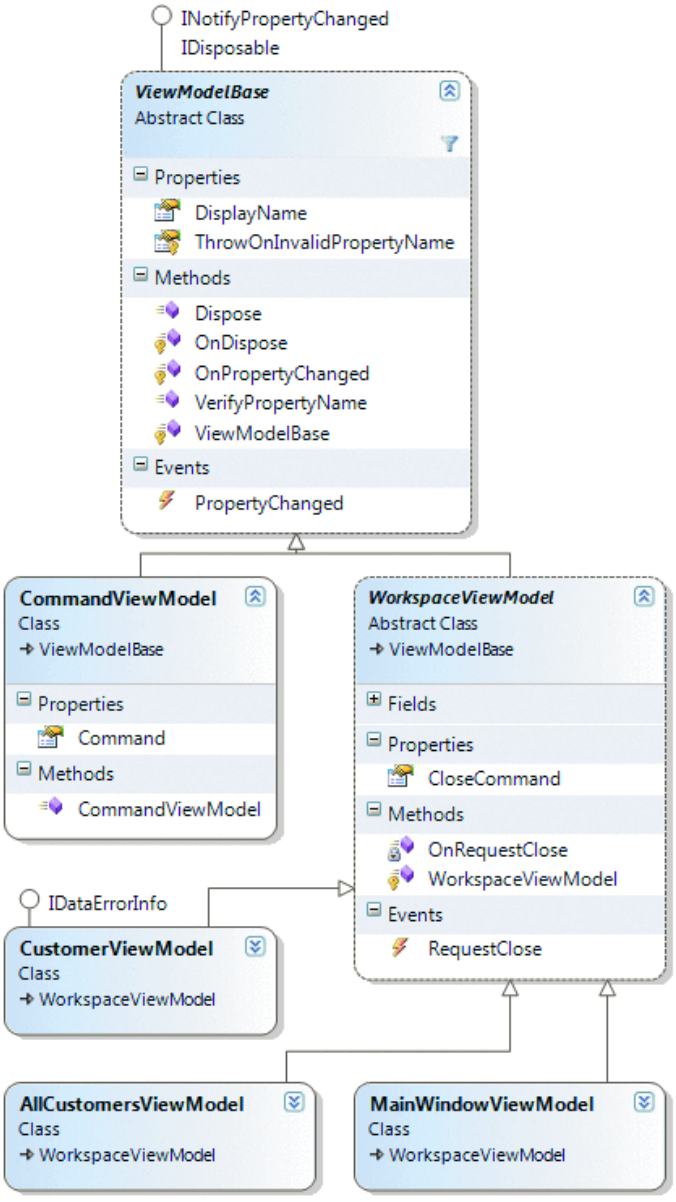

Figura 2.3: La Evolución de Model – View – View Model

A diferencia del moderador en MVP, un ViewModel no tiene una referencia a una vista. La vista se enlaza a las propiedades de un ViewModel, que, a su

vez, expone datos contenidos en objetos del modelo y el otro estado específico a la vista. Los enlaces entre vista y ViewModel son sencillos construir porque un objeto ViewModel está establecido como un DataContext de una vista.

Si cambia los valores de las propiedades del ViewModel, los nuevos valores se propagan automáticamente a la vista a través de enlace de datos. Cuando el usuario hace clic en un botón en la vista, se ejecuta un comando en el ViewModel para realizar la acción solicitada. El ViewModel, realiza todas las modificaciones realizadas en los datos del modelo.

2.7. Historia de los archivos XAML

El XML proviene de un lenguaje que inventó IBM allá por los años 70. El lenguaje de IBM se llama GML (General Markup Language) y surgió por la necesidad que tenían en la empresa de almacenar grandes cantidades de información de temas diversos.

Imaginar por un momento la cantidad de documentación que generaría IBM sobre todas las áreas en las que trabajaba e investigaba, y la cantidad de información que habrá generado hasta hoy. Así pues, necesitaban una manera de guardar la información y los expertos de IBM se inventaron GML, un lenguaje con el que poder clasificarlo todo y escribir cualquier documento para que se pueda luego procesar adecuadamente.

Este lenguaje gustó mucho a la gente de ISO, una entidad que se encarga de normalizar cuantas cosas podáis imaginar para los procesos del mundo actual, de modo que allá por el 86 trabajaron para normalizar el lenguaje, creando el SGML, que no era más que el GML pero estándar.

SGML es un lenguaje muy trabajado, capaz de adaptarse a un gran abanico de problemas y a partir de él se han creado los siguientes sistemas para almacenar información.

Por el año 89, para el ámbito de la red Internet, un usuario que había conocido el lenguaje de etiquetas (Markup) y los hiperenlaces creó un nuevo lenguaje llamado HTML, que fue utilizado para un nuevo servicio de Internet,

la Web. Este lenguaje fue adoptado rápidamente por la comunidad y varias organizaciones comerciales crearon sus propios visores de HTML y riñeron entre ellos para hacer el visor más avanzado, inventándose etiquetas como su propia voluntad les decía. Desde el 96 hasta hoy una entidad llamada W3C ha tratado de poner orden en el HTML y establecer sus reglas y etiquetas para que sea un estándar. Sin embargo el HTML creció de una manera descontrolada y no cumplió todos los problemas que planteaba la sociedad global de Internet.

El mismo W3C en el 98 empezó y continúa, en el desarrollo de XML (Extended Markup Language). En este lenguaje se ha pensado mucho más y muchas personas con grandes conocimientos en la materia están trabajando todavía en su gestación. Pretendían solucionar las carencias del HTML en lo que se respecta al tratamiento de la información. Problemas del HTML como:

- El contenido se mezcla con los estilos que se le quieren aplicar.
- No permite compartir información con todos los dispositivos, como pueden ser ordenadores o teléfonos móviles.
- La presentación en pantalla depende del visor que se utilice.

Imagínese, una persona que conoce el HTML y lo difícil que puede llegar a ser entender su código, que tuviese que procesarlo para extraer datos que necesite en otras aplicaciones. Sería muy difícil saber dónde está realmente la información que busca, siempre mezclada entre etiquetas <FONT>, <TABLE>, <TD>, etc... Esto es una mala gestión de la información y el XML la soluciona.

2.8. Que es XAML?

El lenguaje de marcado de aplicaciones extensibles o XAML es un lenguaje de marcado basado en XML desarrollado por Microsoft. XAML es el lenguaje que se usa para la presentación visual de las aplicaciones desarrolladas con Microsoft Expression Blend, del mismo modo que HTML es el lenguaje que se usa para la presentación visual de las páginas Web.

Un archivo XAML puede ser compilado para obtener un archivo binario XAML .xaml, el cual puede se insertado como un recurso en un ensamblad de

Framework .NET. En el momento de ejecución, el motor del Framework extrae el archivo .xaml de los recursos del ensamblado, se analiza sintácticamente, y crea el correspondiente árbol visual WPF o Workflow.

Cuando se use en Windows Presentation Foundation, XAML es usado para describir interfaces visuales para usuarios.

WPF permite la definición de objetos en 2D y 3D, rotaciones, animaciones y otra variedad de características y efectos. XAML forma parte de Microsoft Windows Presentation Foundation. WPF es la categoría de funciones de Microsoft .NET Framework 3.0 relacionadas con la representación visual de las aplicaciones basadas en Windows y de las aplicaciones cliente basadas en exploradores Web.

Las aplicaciones basadas en WPF se pueden ejecutar en Windows Vista o en versiones anteriores de Windows siempre que esté instalado Microsoft .NET Framework 3.0 (e Internet Explorer 7.0 en el caso de las aplicaciones cliente de exploradores Web).

La estructura de un archivo XAML es el que se muestra a continuación.

```
<Page x:Class="web_logistica.Page_Menu"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Sistema de Logistica" WindowTitle="Sistema de Logistica"
    ShowsNavigationUI="True"
    OverridesDefaultStyle="False" Loaded="Page_Loaded">
```

Es así como inicia un archivo XAML para una aplicación web, como se darán cuenta inicia con el Tag ***Page,*** mientras que para una aplicación de escritorio el inicio del Tag ***Window*** como se indica a continuación.

```
<Window x:Class="web_logistica.Page_Menu"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Sistema de Logistica" WindowTitle="Sistema de Logistica"
    ShowsNavigationUI="True"
    OverridesDefaultStyle="False" Loaded="Page_Loaded">
```

Como se habrán dado cuenta la estructura de los archivos XAML son similares a los de un XML ya que desde allí nació esta nueva tendencia de archivos que son muy potentes por el uso de objetos.

2.9. Cuál es la ventaja de usar los archivos XAML?

Al venir analizando la tecnología WPF nos damos cuenta que existen varias mejoras de las anteriores tecnologías, realizando sistemas de alta presentación, siendo los sistemas agradables a la vista del que lo usa. Entre las principales ventajas de usar archivos XAML tenemos las siguientes:

- Permite la definición de objetos en 2D y 3D, rotaciones, animaciones y otra variedad de características y efectos.
- Permite la creación de temas para los controles, de tal forma que se programa una sola vez y estos son usados para otros controles del mismo tipo.
- Permite obtener la información de forma organizada y entendible ya que se usa la lógica de los archivos HTML, es decir usar tag´s para el manejo de creación de objetos, animaciones, efectos, eventos.
- En estos archivos se pueden hacer referencia a archivos ensamblados .dll y a sus respectivas clases que esta contenga.

Un archivo XAML se comporta como una clase de eventos para los diferentes controles que lo usen, es decir dentro de ella se pueden implementar animación para un botón dándole a cada evento una acción a seguir, por ejemplo si deseamos que el botón a la hora de pasar el mouse sobre el este se resalte como si fuese un foco, se lo puede hacer incluyendo código xml dentro del archivo, este template realizado es usado por el tipo de control programado.

2.10. Análisis de las BDD'S soportadas por la tecnología WPF.

Como lo hemos dicho en varias partes del documento WPF es la evolución de ciertos procesos y tecnologías existentes así que todas las formas de conexión que existía antes con la tecnología de Windows forms lo es posible en WPF,

entre estos tenemos: ADO .NET que viene desde VISUAL .NET 2005 hasta la actualidad. OLEDB (o los conocidos puentes) que viene aun mas tarde que la anterior, esta también es usada por Java para la conexión a base de datos.

WPF implementa otra forma de conexión como es LINQ, esta se conecta usando clases derivadas de la base de datos, actualmente se ha probado que LINQ es bien aprovechado con base de datos SQLSERVER.

Entre las bases de datos que WPF usa tenemos las siguientes:

SQLSERVER.- Base de datos muy potente creado por Microsoft, esta base de datos trabaja muy bien con todas las herramientas de Microsoft como lo es con VISUAL STUDIO .NET.

Fue presentada por Microsoft en el año 2001 como SQL Server CE 1.0 para ofrecer almacenamiento relacional y sincronización de aplicaciones móviles Pocket PC 2002 desarrolladas con Embeded Visual Basic y Embedded Visual C++ con sistemas SQL Server 2000. Junto con la aparición de Visual Studio .NET 2003 y con ello .NET Compact Framework 1.1 se presentó SQL Server CE 2.0 y System.Data.SqlServerCe.dll como interfaz de desarrollo para aplicaciones .NET Compact Framework junto a la interfaz de desarrollo para aplicaciones nativas.

En noviembre del 2006, junto al lanzamiento oficial de Visual Studio .NET 2005 y SQL Server 2005, Microsoft lanza la versión 3.0 renombrada a SQL Server Mobile 3.0. Tras esta versión aparecen nuevas características y funcionalidades como el acceso simultáneo, soporte transaccional o la integración con Visual Studio .NET y SQL Server Management Studio. Hasta la fecha, SQL Server CE no ofrecía ningún tipo de herramientas de administración más que la *Query Analyzer* para Windows Mobile. Meses más tarde se anuncia una nueva versión bajo el nombre en clave SQL Server Every-where la cual ofrecía soporte multiplataforma para sistemas operativos basados en Windows CE y Windows 32/64 bits. Dicha edición fue lanzada oficialmente bajo el nombre SQL Server Compact 3.1.

ORACLE.- Otra base de datos muy potente, su creador IBM. Oracle surge a finales de los 70 bajo el nombre de [Software] a partir de un estudio sobre SGBD (Sistemas Gestores de Base de Datos) de George Koch. Computer World definió este estudio como uno de los más completos jamás escritos sobre bases de datos. Este artículo incluía una comparativa de productos que erigía a Relational Software como el más completo desde el punto de vista técnico. Esto se debía a que usaba la filosofía de las bases de datos relacionales, algo que por aquella época era todavía desconocido.

La tecnología Oracle se encuentra prácticamente en todas las industrias alrededor del mundo y en las oficinas de 98 de las 100 empresas Fortune 100. Oracle es la primera compañía de software que desarrolla e implementa software para empresas 100 por ciento activado por Internet a través de toda su línea de productos: base de datos, aplicaciones comerciales y herramientas de desarrollo de aplicaciones y soporte de decisiones. Oracle es el proveedor mundial líder de software para administración de información, y la segunda empresa de software.

MYSQL.- Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporación desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Michael Widenius en la década de los 90 trató de usar *mSQL* para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y

flexible para sus necesidades. Esto lo llevó a crear una API SQL denominada **MySQL** para bases de datos muy similar a la de *mSQL* pero más portable.

La procedencia del nombre de MySQL no es clara. Desde hace más de 10 años, las herramientas han mantenido el prefijo My. También, se cree que tiene relación con el nombre de la hija del cofundador Monty Widenius quien se llama *My*.

Por otro lado, el nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso "Name the Dolphin". Este nombre fue enviado por Ambrose Twebaze, un desarrollador de software de código abierto africano, derivado del idioma Siswate, el idioma local de Swazilandia y corresponde al nombre de una ciudad en Arusha, Tanzania, cerca de Uganda la ciudad origen de Ambrose.

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones

POSTGRES.- es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

PostgreSQL ha tenido una larga evolución, la cual se inicia en 1982 con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en *Ingres* y de haber tenido una experiencia comercial con él mismo, Michael decidió volver a la Universidad en 1985 para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones, las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario.