

**SEP**

SECRETARÍA DE  
EDUCACIÓN PÚBLICA



## **INSTITUTO TECNOLOGICO DE MEXICALI**

**ANALISIS COMPARATIVO DE LAS TECNOLOGIAS .NET Y  
MONO**

**TESIS  
PARA OBTENER EL GRADO DE  
MAESTRO EN INGENIERIA ELECTRÓNICA**

**PRESENTA  
ING. OMAR DELGADILLO QUEZADA**

**DIRECTOR DE TESIS  
DR. ARNOLDO DÍAZ RAMIREZ**

**CO-DIRECTOR  
MC.JUAN FRANCISCO IBAÑEZ SALAS**



MEXICALI, B.C, ENERO DE 2013

# Análisis Comparativo de las Tecnologías .Net y Mono

# Agradecimientos

Quiero expresar mi agradecimiento a mi familia. También a mi esposa Cristina Trueba por soportar las horas dedicadas a este proyecto. Además a mi asesor Arnoldo Díaz por apoyo incondicional y el tiempo dedicado a este trabajo y demás personas en la institución que participaron en este trabajo gracias.

# Resumen

El presente documento de tesis nace con la finalidad de analizar y comparar los marcos de referencia .Net y mono, para lograrlo se requiere conocer el marco de referencia .Net así como sus principales características. .Net es una solución integral para el desarrollo de aplicaciones de una manera fácil y rápida. A la par de este proyecto y su estandarización se abre la puerta para que el interesado en el pueda realizar una implementación, Miguel de Icaza se intereso tanto por la tecnología .Net que nace el proyecto mono que comenzó como una implementación parcial y termina siendo una implementación casi completa de la tecnología .Net. El trabajo realizado consiste en desarrollar una aplicación utilizando software libre y software propietario, dicha aplicación accede a una base de datos y nos da la opción de realizar altas, bajas y nos muestra los elementos existentes en la base de datos. Finalmente se da a conocer los resultados de dicho análisis comparativo en donde se menciona cual es la dificultad que ofrece cada una de estas herramientas en el proceso de la creación de una solución.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivo general . . . . .	2
1.3. Objetivos específicos . . . . .	2
1.4. Organización del documento . . . . .	2
<b>2. Marco de referencia Microsoft .Net</b>	<b>4</b>
2.1. Marco de referencia .Net de Microsoft . . . . .	6
2.1.1. Componentes Básicos del Marco de Referencia .Net . . . . .	6
2.1.1.1. Common Language Runtime (CLR) . . . . .	7
2.1.1.2. Common Intermediate Language (CIL) . . . . .	7
2.1.1.3. Compilación JIT . . . . .	7
2.1.2. Librería de clases del marco de referencia .Net . . . . .	7
2.2. Componentes de Microsoft .Net . . . . .	8
<b>3. Mono</b>	<b>10</b>
3.1. Historia de Mono . . . . .	11
3.2. Marco de referencia Mono . . . . .	13
3.3. El runtime de Mono . . . . .	15
3.3.1. Mono JIT . . . . .	16
3.3.2. Multi-threading . . . . .	17
3.3.3. PInvoke . . . . .	17
3.4. Compilador C# de Mono . . . . .	18
3.5. Biblioteca de clases externas . . . . .	19
<b>4. Herramientas integrales de desarrollo</b>	<b>20</b>
4.1. NetBeans . . . . .	20
4.2. Eclipse . . . . .	21
4.3. Visual Studio . . . . .	23
4.4. WinDev . . . . .	24
4.5. MonoDevelop . . . . .	25
4.5.1. Características . . . . .	25
<b>5. Trabajo relacionado</b>	<b>27</b>

## ÍNDICE GENERAL

<b>6. Desarrollo de aplicaciones utilizando software propietario</b>	<b>29</b>
6.1. Creación de un nuevo proyecto . . . . .	29
6.2. Creación de la base de datos . . . . .	29
6.3. Creación de la ventana principal . . . . .	32
6.4. Agregar funcionalidad a los botones . . . . .	35
6.5. Aplicación en ejecución . . . . .	38
<b>7. Desarrollo de aplicaciones utilizando software libre</b>	<b>41</b>
7.1. Creación de un nuevo proyecto . . . . .	41
7.2. Creación de la base de datos. . . . .	41
7.3. Creación de la ventana principal . . . . .	44
7.4. Agregar funcionalidad a los botones . . . . .	53
7.5. Aplicación en ejecución . . . . .	55
<b>8. Análisis comparativo</b>	<b>57</b>
<b>9. Conclusiones y trabajo futuro</b>	<b>59</b>

# Índice de figuras

2.1. Marco de referencia .Net . . . . .	6
3.1. Marco de referencia Mono . . . . .	13
3.2. Implementación en Mono del fractúrame .ten . . . .	15
3.3. Compilación de JIT . . . . .	17
3.4. Compilador C# de Mono . . . . .	18
4.1. IDE NetBeans v7.0.1 . . . . .	22
4.2. IDE Eclipse v3.7.2 . . . . .	23
4.3. IDE Visual Studio 2012 . . . . .	24
4.4. MonoDevelop v2.8.6.3 . . . . .	26
6.1. Pantalla inicial de visual studio 2012 . . . . .	30
6.2. Creando aplicación windows form . . . . .	30
6.3. Aplicación windows form . . . . .	31
6.4. Nombrando base de datos . . . . .	31
6.5. Agregar nueva tabla a la base de datos . . . . .	32
6.6. Diseño de la tabla usuarios . . . . .	32
6.7. Agregando elementos la tabla . . . . .	33
6.8. Creando DataGridView para mostrar base de datos . . . . .	33
6.9. Selección del origen de los datos . . . . .	34
6.10. Elegir la conexión de datos . . . . .	34
6.11. Selección de tabla y datos a mostrar . . . . .	34
6.12. Agregando label1 . . . . .	35
6.13. Agregando TextBox1 . . . . .	36
6.14. Agregando button1 . . . . .	36
6.15. Agregando label3, TextBox3 y button2 . . . . .	37
6.16. Ventana terminada . . . . .	37
6.17. Agregando usuario . . . . .	39
6.18. Eliminando usuario . . . . .	40
7.1. Crear nueva solución . . . . .	42
7.2. Selección de proyecto GTK . . . . .	42
7.3. Selección de GTK apropiado . . . . .	43

## ÍNDICE DE FIGURAS

---

7.4. Diseño inician del proyecto GTK . . . . .	43
7.5. Creación de base de datos MySQL . . . . .	44
7.6. Nombrando la Base de Datos . . . . .	45
7.7. Finalizando la creación del esquema ITMDB . . . . .	45
7.8. Creando Tabla usuarios en MySQL . . . . .	46
7.9. Nombrando Tabla en MySQL . . . . .	46
7.10. Finalizando la ceracion de la Tabla usuarios . . . . .	47
7.11. Insertando datos a la tabla usuarios . . . . .	47
7.12. Agregando VBox a MainWindows.cs . . . . .	48
7.13. Agregando Scrolled windows a VBox . . . . .	49
7.14. Agregando Tree View en Scrolled window . . . . .	49
7.15. Agregando HBox en VBox . . . . .	50
7.16. Agregando nuevo contenedor a HBox . . . . .	50
7.17. Agregando Label1 . . . . .	51
7.18. Agregando Entry1 . . . . .	51
7.19. Agregando Button1 . . . . .	52
7.20. Agregando Button2 . . . . .	52
7.21. Ventana terminada . . . . .	53
7.22. Agregando evento al botón Agregar . . . . .	55
7.23. Agregando al usuario . . . . .	56
7.24. Eliminando usuario . . . . .	56

# Índice de tablas

3.1. Porcentaje de implementación de las clases en 2003 . . . . .	14
8.1. Dificultad presentada en las tecnologías .Net y Mono . . . . .	58

# Capítulo 1

## Introducción

### 1.1. Motivación

El mundo del desarrollo de aplicaciones se encuentra sumido en una nueva etapa de transformación y evolución hacia nuevos esquemas de trabajo. Los factores determinantes de dicho cambio los podemos encontrar en la necesidad de utilizar internet como vehículo de intercambio por parte de diversos sectores de la economía, y a la gran proliferación de dispositivos móviles que han ido emergiendo en los últimos años.

Las empresas al ver estas nuevas necesidades se dan cuenta que se requieren relaciones comerciales mas dinámicas con sus clientes. De modo que su volumen de negocio se incremente a través del canal de ventas electrónico. Por otro lado también necesitan relaciones empresariales mas ágiles en este marco de ciberespacio.

Aparte de todos estos elementos, nos encontramos con que los usuarios de este medio, disponen de dispositivos cada vez mas sofisticados para desplazarse por la red, no solo las PC o laptops, en la actualidad se cuenta con smartphones, tablets, etc,. Una gran variedad de dispositivos móviles que permiten un acceso rápido y sencillo, a múltiples aplicaciones simultáneamente desde cualquier lugar en el que tengamos acceso a internet, todo esto con un mayor grado de interacción, y obteniendo información de un amplio conjunto de fuentes de datos, y esto sin los esfuerzos de configuración que requerían algunas aplicaciones antiguas.

Debido a todo lo anterior surgió la necesidad de nuevas herramientas que facilitaran en el diseño y acceso a bases de datos y dejaran que el programador solo se encargara de la lógica del programa. Desde 1984, PC SOFT ha estado trabajando por distribuir herramientas de alta tecnología accesibles al número más grande posible de personas.

El primer “best seller” de PC SOFT fue un generador de pantallas llamado HI SCREEN (10,000 copias vendidas en USA). Basados en la fortaleza de la experiencia que se ganó de esta herramienta que vendió decenas de miles de copias a nivel mundial, PC SOFT creó WINDEV, un IDE para Windows que era el

más completo y el más fácil de usar. A finales del año 2000 .Net surge como un marco de referencia de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basada en todos sus productos, desde el sistema operativo hasta las herramientas de mercado que esta ofrece. la mayor parte de .Net estaba estandarizado y por tanto Miguel de Icaza se intereso en la implementación de una plataforma libre que incorporase compiladores, runtimes y librerías de clases. Pero Miguel tenía dos grandes obstáculos con .Net: no era gratuito y no trabajaba sobre Linux. Con estos problemas, de Icaza y su equipo empezó la construcción de su propia versión de .Net.

El proyecto Mono lanzó su primer versión de su software, basado en C# y en los componentes estándar CLI (Common Language Infrastructure) de .Net y añadidas a los estándares ECMA e ISO. Ambos, .Net y Mono proveen los mismos API's, los cuales pueden ser llamados desde múltiples lenguajes. Ellos también proveen integración simplificada con los lenguajes, y el CLR (Common Language Runtime), el cual es similar al JVM (Java Virtual Machine) de Java. Conociendo todo esto nos dimos a la tarea de evaluar estas dos herramientas para conocer las facilidades que ofrecen las herramientas de para la creación de aplicaciones libres contra las herramientas propietarias.

## 1.2. Objetivo general

Realizar un análisis comparativo entre las tecnologías .Net de tal manera que se conozcan las ventajas y desventajas de las principales características que ofrecen los marcos de referencia de software libre (Mono) y software propietario (Microsoft).

## 1.3. Objetivos específicos

- Describir las características mas importantes del marco de referencia .Net.
- Desarrollo de soluciones .Net utilizando software propietario Microsoft.
- Desarrollo de soluciones .Net utilizando software libre.
- Análisis comparativo del desarrollo de aplicaciones .Net utilizando software libre y software propietario.

## 1.4. Organización del documento

El presente trabajo de tesis esta organizada de la siguiente manera, en el capítulo 2 se hablara de la tecnología Microsoft .Net, se explicara que es el marco de referencia .Net, como es que funciona y algunas de sus características.

Se abordara el panorama actual acerca de los entornos de desarrollo mas utilizados en la actualidad y daremos una breve reseña de cada uno.

## CAPÍTULO 1. INTRODUCCIÓN

En el capítulo 3 se menciona que es la tecnología Mono, cuales son sus orígenes, como funciona su runtime y el compilador C# de mono y se mencionan algunas de las librerías extras que se pueden integrar a esta tecnología.

En el capítulo 4 se mencionan algunas herramientas integrales de desarrollo mas conocidas.

En el capítulo 5 en esta sección mencionamos algunos artículos relacionados con el presente trabajo.

En el capítulo 6 se realiza el desarrollo de aplicaciones utilizando software libre paso a paso.

En el capítulo 7 Se realiza el desarrollo de aplicaciones utilizando software propietario paso a paso.

Finalmente en el capítulo 8 se concluyen los resultados de dicha comparación y da a conocer el trabajo que se podría realizar en el futuro.

## Capítulo 2

# Marco de referencia Microsoft .Net

.Net es la plataforma de Microsoft para el desarrollo, despliegue y ejecución de aplicaciones orientadas a servicios sobre entornos altamente distribuidos, tal y como lo es Internet. Además introduce el concepto de los servicios Web, que permiten el desarrollo de aplicaciones acopladas basadas en componentes que utilizan protocolos de comunicación estándares de Internet como SOAP [18] (Simple Object Access Protocol) y XML [17] (Extensible Markup Language).

La plataforma .Net no es un sistema operativo, al menos por el momento, si bien está bastante integrada con este, y hace uso de los servicios que le proporciona. El estado actual de .Net podría compararse con el estado del entorno Windows 3.1 con respecto al sistema operativo MS-DOS [20] (Microsoft Disk Operating System), por lo que es más que probable que la plataforma .Net acabe fundiéndose con una futura versión del sistema operativo Windows.

La plataforma .Net en realidad no es algo radicalmente nuevo. Es un conjunto de tecnologías dispersas, que en muchos casos ya existían, y que Microsoft ha integrado en una plataforma común con el objetivo de facilitar el desarrollo de este nuevo tipo de servicios de tercera generación.

.Net representa la visión del software como un servicio, habiendo sido diseñada con Internet en mente. Esta plataforma cubre todas las capas del desarrollo de software, existiendo una alta integración entre las tecnologías de presentación, de componentes y de acceso a datos. Intenta poner un cierto orden sobre el caos del desarrollo de aplicaciones distribuidas, que se basaba en un modelo de tres capas, con ASP [13](Active Server Pages) en la capa de presentación, COM [19] en la capa de objetos de negocio y ADO [8](Access Data Objects) en la capa de datos; dicha plataforma tenía como problemas principales que el desarrollo con COM era complejo y poseía una integración con ASP un tanto artificiosa.

La plataforma .Net ha sido diseñada con la intención de satisfacer los siguientes objetivos:

- Proporcionar un modelo de programación simple y consistente. A difer-

## CAPÍTULO 2. MARCO DE REFERENCIA MICROSOFT .NET

encia de los modelos ya pasados, en los cuales algunas facilidades del sistema operativo son ofrecidas mediante DLL's y otras mediante objetos COM, todos los servicios de Mono son proporcionados de la misma forma mediante un modelo de programación orientado a objetos. Así mismo, se ha simplificado el modelo de programación, lo que permite a los desarrolladores centrarse en las cuestiones relativas a la lógica de la aplicación; se ha eliminado la necesidad de generar ficheros IDL, gestionar el registro, etc.

- Liberar al programador de las cuestiones de infraestructura (aspectos no funcionales). Así, .Net se encarga de gestionar automáticamente tales cuestiones como la gestión de la memoria, de los hilos o de los objetos remotos.

- Proporcionar integración entre diferentes lenguajes. Con el auge de los sistemas distribuidos, la interoperabilidad se ha convertido en una de las principales cuestiones de los desarrolladores de sistemas. El problema de la interoperabilidad ha sido considerado durante muchos años, desarrollándose varios estándares y arquitecturas como son los estándares arquitecturales (RPC – Remote Procedure Calling, CORBA - Common Object Request Broker Architecture, COM, los estándares de lenguajes ANSI C, etc.).

- Proporcionar una ejecución multiplataforma. .Net ha sido diseñado para ser independiente de la plataforma sobre la cual se ejecutarán las aplicaciones. Para conseguir este objetivo las aplicaciones .Net se compilan a un lenguaje intermedio denominado Lenguaje Común Intermedio (CIL, Common Intermediate Language), el cual es independiente de las instrucciones de una CPU concreta.

- Sistema de despliegue simple. Se ha eliminado la necesidad de tratar con el registro, con GUIDs (cadena de identificación única usada con llamadas a procedimientos remotos), etc., de forma que la instalación de una aplicación es tan sencilla como su copia en un directorio.

- Mejora de la escalabilidad. La gestión por parte del sistema de ejecución de .Net de cuestiones como la memoria permite mejorar la escalabilidad.

- Proporcionar un mecanismo de seguridad avanzado. El aumento de la dependencia sobre el código móvil, como los scripts Web, la descarga de aplicaciones de Internet o los correos con binarios adjuntos, ha provocado que el modelo tradicional de seguridad basado en cuentas de usuario haya dejado, en parte, de tener sentido, pues asume que todo el código, ya sea móvil o no, tiene el mismo nivel de confianza. Así, la plataforma .Net proporciona un modelo de seguridad basado en la evidencia, que posee un modelo de control de gran granularidad, pudiendo basarse o no en quien escribió el código, que intenta hacer dicho código, donde está instalado, y quien está intentando ejecutar dicho código.

Con estos objetivos, Microsoft .Net es una plataforma para construir, ejecutar y experimentar la tercera generación de aplicaciones distribuidas, que consiste en los siguientes elementos:

1. Un modelo de programación basado en XML(Extensible Markup Language).

2. Un conjunto de servicios Web XML para facilitar a los desarrolladores integrar estos servicios.

3. Un conjunto de servidores que permiten ejecutar estos servicios.

## CAPÍTULO 2. MARCO DE REFERENCIA MICROSOFT .NET

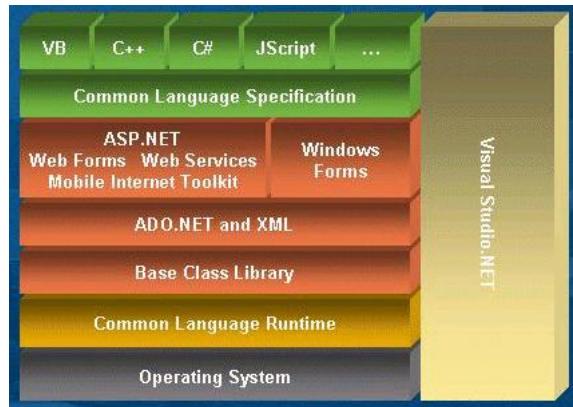


Figura 2.1: Marco de referencia .Net

4. Software en el cliente para poder utilizar estos servicios (como Windows XP y posteriores , ipad, smartphones, etc.).

5. Herramientas para el desarrollo

.Net se encuentra dentro de un entorno en el cual hay muchos más productos y aplicaciones, y que en este caso, a diferencia de casi todos sus productos anteriores, Microsoft ha abierto hasta cierto punto su entorno, de forma que todo el mundo pueda participar en él. El entorno dentro del que se encuadra .Net es una Internet que están cambiando de ser centrada en las personas, y basada en los contenidos, a estar centrada en las aplicaciones, y basada en los servicios. Estas aplicaciones y servicios forman parte de lo que se llaman servicios Web.

### 2.1. Marco de referencia .Net de Microsoft

El marco de referencia Microsoft .Net es un importante componente en la familia de sistemas operativos de Microsoft Windows. Es la infraestructura de la plataforma Microsoft .Net. Es un ambiente común para la siguiente generación de aplicaciones que son fáciles de construir, desarrollar e integrar con otros sistemas en una red. La Figura 2.1 representa el marco de referencia .Net de microsoft, y esta se describe brevemente a continuación.

#### 2.1.1. Componentes Básicos del Marco de Referencia .Net

El marco de referencia .Net consiste en dos partes principales: El Common Language Runtime (CLR) y la librería de clases de marco de referencia .Net.

### **2.1.1.1. Common Language Runtime (CLR)**

El motor de ejecución del CLR (Common Language Runtime) es el responsable de asegurar que el código es ejecutado como requiere, proporcionando una serie de facilidades para el código CIL (Common Intermediate Language) como:

- Carga del código y verificación.
- Gestión de las excepciones.
- Compilación “Just In Time” (JIT).
- Gestión de la memoria.
- Seguridad.

### **2.1.1.2. Common Intermediate Language (CIL)**

El código intermedio CIL conocido como el estándar ECMA-335 [12] generado por los compiladores del marco de referencia .Net es independiente del juego de instrucciones de una CPU específica, pudiendo ser convertido a código nativo de forma eficiente. El lenguaje CIL es un lenguaje de un nivel de abstracción mucho mayor que el de la mayoría de los lenguajes máquina de las CPUs existentes, incluyendo instrucciones para trabajar directamente con objetos (crearlos, destruirlos, inicializarlos, llamar a métodos virtuales, etc.), instrucciones para el manejo de excepciones, de tablas, etc.

La principal ventaja del CIL es que proporciona una capa de abstracción del hardware, lo que facilita la ejecución multiplataforma y la integración entre lenguajes. Otra ventaja que se deriva del uso de este lenguaje intermedio es la cuestión de la seguridad relativa a la verificación del código, pues el motor de ejecución puede examinar la intención del código independientemente del lenguaje de alto nivel utilizado para generararlo. Sin embargo, dado que las CPUs no pueden ejecutar directamente CIL, es necesario convertirlo a código nativo de la CPU antes de ejecutarlo.

### **2.1.1.3. Compilación JIT**

La traducción de CIL a código nativo de la CPU es realizada por un compilador “Just In Time” o jitter, que va convirtiendo dinámicamente el código CIL a ejecutar en código nativo según sea necesario.

## **2.1.2. Librería de clases del marco de referencia .Net**

La librería de clases incluye un conjunto de paquetes de gran funcionalidad que los desarrolladores pueden usar para extender más rápidamente las capacidades de su propio software. La librería incluye tres componentes claves:

1. ASP.Net para ayudar a construir aplicaciones Web y servicios Web
2. Windows Forms para facilitar el desarrollo de interfaces de usuario para clientes inteligentes
3. ADO.Net para ayudar a conectar a las aplicaciones con las bases de datos.

## **2.2. Componentes de Microsoft .Net**

El componente principal de .Net, que está en la capa más baja de su modelo de capas, es el CLR (Common Language Runtime), o máquina virtual común. Se trata de un programa, que se puede ejecutar, en principio, en cualquier sistema operativo, y que provee de una serie de servicios que se pueden usar desde diferentes lenguajes de programación.

Hay implementaciones no basadas en el código de Microsoft; la principal es el proyecto Mono, de la empresa Ximian[2]. La especificación de este CLR se quiere convertir en un estándar ECMA (European Computer Manufacturers Association), de forma que pueda haber diferentes implementaciones de la misma, y diferentes lenguajes basados en ella.

Los ejecutables CLR (Common Language Runtime), están escritos en un lenguaje denominado MSIL (Microsoft Intermediate Language), similar al Java y su bytecode; en principio, cualquier programa escrito en MSIL (aunque nadie escribe en MSIL, se supone que lo hacen los compiladores) puede ejecutarse en cualquier sistema operativo donde funcione un CLR; el formato de esos archivos se denomina PE (Portable Executable). Además, el fichero ejecutable contiene metadatos, que informan sobre las funciones y tipos que implementan. Pero el concepto de ejecutable va un poco más allá: en .Net se usan ensamblajes, que pueden incluir partes de código, datos, códigos de seguridad, y todo lo necesario para convertirlo en código móvil y fiable (en el sentido de que esté firmado por alguien), que se pueda mover por Internet.

Los ensamblajes, a su vez, contienen metadatos, igual que sucede en los .jar de Java. Estos ejecutables y ensamblajes pueden ser generados a partir de diferentes lenguajes de alto nivel, pero los lenguajes deben de incluir dos cosas: un sistema común de tipos (CTS, Common Type System) y un sistema común de lenguajes (CLS, Common Language Specification).

El sistema común de tipos indica los tipos que tiene que soportar el lenguaje: tipos valor (por ejemplo, un entero; una variable entera contiene un valor) y tipos referencia, que apuntan a estructuras de datos dinámicas. Sin embargo, a diferencia de los lenguajes habituales, donde el tipo fundamental es un tipo valor, y las referencias son accesorias, y deben desreferenciarse para trabajar con ellas, en .Net el tipo fundamental es un objeto, y, de hecho, cualquier tipo de valor se puede convertir en una referencia "encajándolo" (boxing).

El CTS (Common Type System) añade soporte para una serie de tipos que no se suelen encontrar en otros lenguajes: eventos (métodos que responden a un suceso determinado), propiedades (métodos para establecer y recuperar valores de variables de instancia) e indexadores, similares a los iteradores usados en otros lenguajes.

En cuanto a la especificación común de lenguaje (CLS, Common Language Specification), son una serie de reglas básicas requeridas para integración del lenguaje, que garantice que el código intermedio generado desde cada uno de ellos sea interoperable con los otros. Hasta ahora, hay una serie de lenguajes propios de Microsoft: J# (similar al Java), VB.Net, Perl.Net, Python.Net. El más popular probablemente es C#, al que efectivamente, se le parece, pero

## CAPÍTULO 2. MARCO DE REFERENCIA MICROSOFT .NET

que es un lenguaje totalmente diferente, con bastantes cosas originales. Todos los lenguajes usan el mismo conjunto básico de servicios, proporcionados por el CLR (Common Language Runtime): entrada salida, acceso al sistema de archivos, acceso a servicios remotos y acceso a datos.

El XML (Extensible Markup Language) está totalmente integrado con el C#: un programa permite pasar la definición de una clase a un fichero XML en formato XSchema (un formato que permite especificar, a su vez, el formato en el que se tiene que escribir un documento).

Hay una serie de problemas en este entorno: la disponibilidad del código fuente, ya que el MSIL (Microsoft Intermediate Language) se puede desensamblar con relativa facilidad, y además, tiene los metadatos que complementan más todavía la legibilidad del código; por eso, en caso de que no se quiera publicar el código, hay que hacer uso de algún tipo de técnica de enmascaramiento.

El siguiente problema son las prestaciones, problema común a todo tipo de máquina virtual; sin embargo, con el compilador JIT (Just in Time, similar al que tienen las máquinas virtuales Java), se trata de obtener el máximo rendimiento del código, compilándolo sólo cuando se le cargue por primera vez. Y, por supuesto, sobre todo esto está la sombra de la historia pasada de Microsoft: tener un estándar cerrado, que puede ser cambiado arbitrariamente, lo cual puede dejar fuera del negocio a muchos.

Otro problema adicional es la falta de servicios de autenticación. Inicialmente, se iba a usar Passport, que luego se convirtió en Hailstorm, para acabar siendo My Services. Finalmente, por falta de apoyo por parte de la industria, Microsoft decidió suprimirlo. Nadie quería, como es natural, que fuera Microsoft quien autentificara a sus clientes, por mucho que sea Microsoft.

Otro posible problema son los virus; como cualquier formato ejecutable, el PE (Portable Executable) se puede infectar con virus, y si el CLR (Common Language Runtime) se convierte en ubicuo, puede tener bastantes posibilidades de propagación.

Los demás componentes de .Net permiten extender a todos los productos de Microsoft la funcionalidad de .Net:

- ASP.Net: Active Server Pages, en su versión para .Net.
- VB.Net: versión para el CLR del Visual Basic, el lenguaje común a todas las aplicaciones de Microsoft.
- ADO.Net, acceso a objetos de datos ( Access to Data Objects), que permite acceder de forma orientada a objetos a bases de datos; también da una serie de servicios para acceso a bases de datos y otros repositorios de objetos desde dentro de la CLR.
- Perl.Net, Python.Net son desarrollos de ActiveState, que se integran con el entorno Visual Studio .Net y permiten desarrollar programas en esos lenguajes.
- WinForms, diseño gráfico de ventanas dentro de .Net. En la implementación Mono, se sustituye por Gtk#.

# Capítulo 3

## Mono

A finales del 2000 Microsoft publica los primeros documentos sobre la tecnología .Net. En estos se especificaba el funcionamiento de esta nueva plataforma que nacía entre otros motivos para hacer frente al éxito de Java de la competidora Sun.

La idea de .Net tiene bastantes similitudes con la tecnología Java, ambos compilan el código fuente a un código intermedio (no directamente a código máquina). En el caso de Java este código es llamado bytecode y en .Net recibe el nombre de CIL (Common Intermediate Language).

Para ejecutar este código intermedio es necesario un entorno que lo interprete y así poder pasar al código máquina correspondiente al sistema/arquitectura donde se este ejecutando. De esta forma se consigue independencia del ejecutable en contraposición al tradicional compilado a código máquina, ya que este último solo podría ser utilizado en máquinas que soporten el mismo conjunto de instrucciones y en sistemas que conozcan el formato de ese ejecutable.

Pero .Net va más allá, su objetivo no es sólo la independencia del compilado sino también la independencia del lenguaje de alto nivel, es decir, CIL ha sido especialmente diseñado para proporcionar todo lo necesario a la mayoría de lenguajes actuales. El lenguaje que aprovecha toda la potencia de CIL es C# diseñado por la propia Microsoft, pero esto no impide que todo aquel que quiera formar parte de la plataforma .Net construya un compilador de su lenguaje a código intermedio CIL.

Esto nos proporciona por ejemplo la posibilidad de poder reutilizar clases programadas en lenguaje C# desde Visual Basic.Net de forma muy sencilla, cosa que hasta el momento sólo era posible mediante complejos mecanismos poco flexibles y que ahora es posible tener de forma nativa a la plataforma.

.Net tiene definido un CTS (Common Type System) con los tipos de datos soportados, los cuales son suficientes para cubrir cualquier lenguaje actual. Pero esto no es suficiente para garantizar la interoperabilidad entre lenguajes, porque por ejemplo imaginemos que hay un lenguaje que soporta el tipo entero sin signo y otro que no, esto impediría la interoperabilidad entre ambos lenguajes. Por ese motivo también se ha definido el CLS ( Common Language Specification),

el cual es necesario que cumplan todos los lenguajes que quieran poder disfrutar de dicha interoperabilidad.

Esta es la gran diferencia básica con respecto a Java de Sun, ya que el bytecode no ha sido diseñado para tales prácticas (compilación de cualquier lenguaje actual a bytecode) y por tanto no resulta óptima para desempeñar las tareas que puede cubrir CIL.

### 3.1. Historia de Mono

En Diciembre del 2000 Miguel de Icaza (Co-fundador de la empresa Ximian, fundador y presidente de la GNOME Foundation) se interesó bastante por la tecnología .Net al tener acceso a los primeros documentos de Microsoft.

GNOME (GNU Network Object Model Environment) siempre había luchado por proporcionar facilidades al programador y una de las características de más conocidas es que existen multitud bindings (adaptadores) para poder utilizar cualquier lenguaje para desarrollar aplicaciones. Pero la elaboración de dichos bindings era tremadamente laboriosa y cada vez que se realizaba un cambio en la interfaz original, era necesario cambiar todos y cada uno de los bindings.

Para intentar mejorar y facilitar la reutilización de código se realizó una implementación de componentes utilizando CORBA llamada Bonobo. Pero tampoco ha tenido éxito ya que era necesario que todo el mundo utilizase esa característica y eso no fue así.

Por tanto, con .Net se abrió una nueva puerta para conseguir hacer de GNOME en un futuro, un escritorio mejor y más atractivo tanto para usuarios como para programadores. Con esta tecnología por fin se consiguió lo que el proyecto GNOME siempre había buscado, independencia del lenguaje para programar en dicho escritorio.

Además de estos beneficios, la mayor parte de .Net estaba estandarizado y por tanto era viable la implementación de una plataforma libre que incorporase compiladores, runtimes y librerías de clases. Pero Miguel tenía dos grandes obstáculos con .Net: no era gratuito y no trabajaba sobre Linux. Con estos problemas, de Icaza y su equipo empezó la construcción de su propia versión de .Net.

Cuando Mono lanzó la nueva versión de su software, basado en C# y en los componentes estándar CLI (Common Language Infrastructure) de .Net y añadidas a los estándares ECMA (European Computer Manufacturers Association) e ISO (International Organization for Standardization), Microsoft publicó y abrió los API's para .Net. Este fue el mayor paso de Microsoft desde que se lanzó la tecnología, y abrió la puerta a otros para la construcción de implementaciones compatibles del API .Net.

Y la gente empezó a cruzar por esa puerta. El proyecto Mono, liderado por Miguel de Icaza, y Microsoft .Net empezaron a diseñar una implementación de la plataforma de desarrollo .Net. Ambos, .Net y Mono proveen los mismos API's, los cuales pueden ser llamados desde múltiples lenguajes. Ellos también proveen integración simplificada con los lenguajes, y el CLR (Common Language

Runtime), el cual es similar al JVM (Java Virtual Machine) de Java.

A de Icaza no le importó que el y su equipo hayan seguido el liderazgo de Microsoft. Y efectivamente mucha gente estuvo y está todavía interesada en .Net como tecnología, y no les importa que esta tecnología haya sido inventada por Microsoft. Cualquier tipo de problemas se resuelven directamente por .Net, dijo. Y GNOME era un buen lugar para lo que .Net estaba tratando de resolver. El equipo del proyecto Mono buscaba soluciones y encontraron en .Net lo necesario para ello.

La respuesta de Microsoft al código abierto fue menos delicada y más combativa. Pero últimamente los mira como la relación que se está envolviendo en una balance productivo.

Icaza hizo un comentario en el cual Microsoft planeaba compartir la propiedad intelectual relacionada a .Net. En espera de la revisión presentada por los abogados de Microsoft, dijo: "Microsoft patenta sobre el desarrollo de la tecnología, específicamente para .Net; y estará de acuerdo con aquellos que traten de implementarlo".

Esta fue una gran noticia, desde que hubo una discusión en sitios Web de código abierto, en los cuales se comentaba que patentar y poner una propiedad intelectual a este asunto podría ser usado por Microsoft para frustrar cualquier implementación no .Net, a pesar de la presentación de .Net como un estándar.

Microsoft fue más cauteloso en este asunto. John Montgomery, director de la Administración de Producto de Microsoft, dijo que Microsoft seguirá todas las políticas de estandarización ECMA e ISO. Esto especificaba algo "razonable y no discriminatorio" a través de una política de patente no necesariamente libre. Cuando se habló de "razonable y no discriminatorio", se sabía que no mataría al proyecto Mono, simplemente tendría un efecto desalentador fuera del mundo Windows.

En un nivel de ingeniería, la relación entre el equipo del proyecto Mono y la gente de Microsoft fue amigable. Se encontraban en reuniones del ECMA y otros eventos. Era una simpatía natural, debido a que ambos trabajan en cosas similares, las mismas herramientas, y ambos interesados en lo que sucedía con el mundo .Net.

Montgomery tuvo afectos similares. Sostuvo que el hecho de que Ximian como empresa estaba haciendo este trabajo era muy gratificante. Una validación del trabajo que había hecho la gente de Microsoft.

Un efecto de todo esto fue, que los desarrolladores que no habían considerado a .Net, estaban próximos a probarlo, pues conocer sus productos tenía un potencial al desarrollo en más de una plataforma. Esto no necesariamente significaba perdidas en las ventas para Microsoft. En efecto, esto expandía la posibilidad a la oportunidad de Microsoft de vender Visual Studio .Net a un nuevo grupo de desarrolladores, también iba a hacer posible el tener un trabajo en el desarrollo utilizando Visual Studio .Net, y después ponerlo a producir en un servidor .Net o en Mono en Linux, como un cliente o como la situación lo demande.



Figura 3.1: Marco de referencia Mono

### 3.2. Marco de referencia Mono

Mono implementa las siguientes partes de la tecnología .Net:

- Common Language Runtime
- Compilador/Desensamblador IL
- Compilador C#
- Compilador Visual Basic.Net
- Librería de clases (FCL)
- Otras librerías de funcionalidades

Para entender de mejor manera lo que es el marco de referencia mono ver la Figura 3.1.

Los compiladores proporcionados están bajo la licencia GNU GPL, el runtime tiene licencia GNU LGPL y la librería de clases la MIT X11 License. Mono se puede ejecutar en los siguientes sistemas: Plataforma MS Windows Mac OSX FreeBSD GNU/Linux (x86) GNU/Linux (PPC) GNU/Linux (S390) A fecha el compilador de C# y el runtime se encuentran en un estado muy avanzado, ya se puede utilizar perfectamente para cualquier proyecto. El compilador de Visual Basic.Net aún necesita crecer. La librería de clases esta en proceso aunque ya es suficientemente madura y útil como para poder ser utilizada exceptuando Windows.Forms (aplicaciones gráficas, aunque existe un proyecto en muy buen estado llamado GTK# con el cual podemos realizar aplicaciones utilizando las librerías gráficas GTK) y EnterpriseServices.

Algunos componentes más pequeños y menos usados no están siendo desarrollados como son el caso de System.Management y System.Drawing.Design, la Tabla 3.1 muestra el porcentaje de implementación de las clases del marco de referencia .Net en el año 2003.

En cuanto a la implementación .Net de Microsoft solo cubre sus sistemas MS Windows además de Mac OSX y FreeBSD. Ofrecen el código del Common

Cases	% implementado
Corlib(Core Library)	90
System	95
System.XML	97
System.Data	83
System.Drawing	73
System.Web	96
System.Web.Services	96
Microsoft.VisualBasic	68
System.Windows.Forms	50
System.EnterpriseServices	52
System.Runtime.Serialization.Formater.Soap	60
System.Security	93

Tabla 3.1: Porcentaje de implementación de las clases en 2003

Language Infrastructure, es decir, el Runtime y el compilador de C# entre algunas otras pequeñas utilidades pero no toda la implementación de la librería de clases. Este código es ofrecido bajo la licencia Microsoft Shared Source CLI, C#, y jscript License, la cual limita completamente al usuario impidiéndole realizar ninguna modificación y mucho menos redistribuirlo. Del acceso a dicho código podrían incluso derivarse problemas legales si se llegase a inspirar algún programador en el código mostrado.

En cambio Mono ofrece todo su código, compiladores, runtime, librería de clases... con unas licencias libres, perfectas para poder aprender sin preocuparse de posibles problemas legales futuros y con la posibilidad de realizar modificaciones en el código para colaborar o distribuirlo ya sea comercialmente o de forma gratuita.

Mono también en el 2003 pretendía implementar las partes de .Net que no están estandarizadas como ADO.Net (acceso a Bases de datos), ASP.Net (desarrollo de web) y Windows.Forms (aplicaciones gráficas). De esta forma también se puede conseguir una sencilla migración de la plataforma .Net implementada por Microsoft a Mono.

Actualmente Mono cubre gran porcentaje del marco de referencia .Net de Microsoft como se muestra en la Figura 3.2.

En caso de que Microsoft incumpliese el estándar en su implementación se originaría una situación donde el marco de referencia .Net de MS y Mono serían incompatibles, pero Mono aun seguiría siendo útil por si mismo. En el caso de que se hagan patentes sobre las partes no estandarizadas, Mono puede optar por varias soluciones:

- 1) Intentar evitar los problemas de la patente realizando una implementación diferente de la funcionalidad de forma que no se vea afectada la API.
- 2) Eliminar las partes afectadas por la patente.

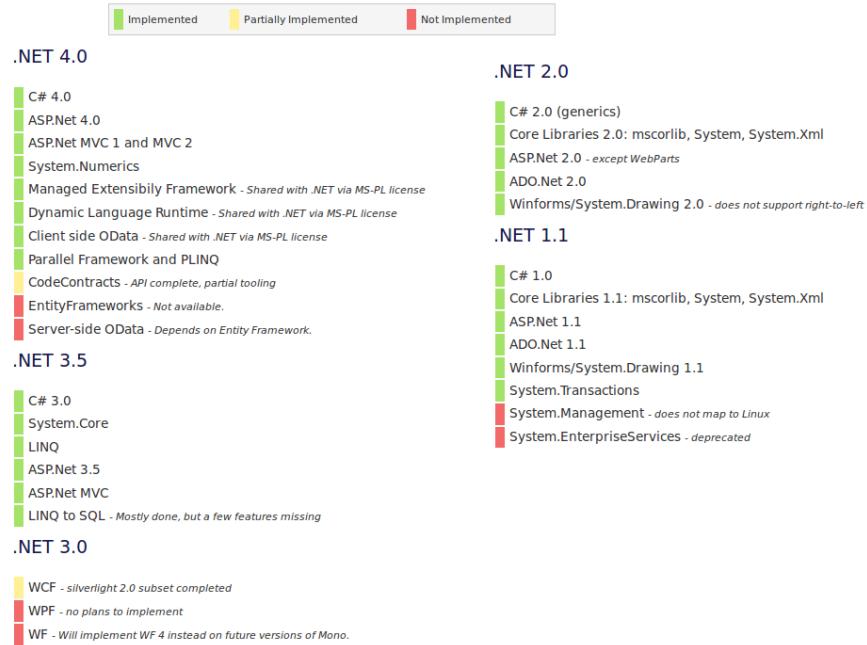


Figure 3.2: Implementación en Mono del fractúrame .ten

3) Encontrar ejemplos de implementaciones anteriores que hagan invalida la patente (e.g. Muchas de las funcionalidades de la librería de clases ya existían antes en Java y por tanto esto invalidaría una patente).

A todo esto también cabe destacar que aquellos países donde las patentes de software no sean válidas no se verían afectados por estas posibles acciones.

### 3.3. El runtime de Mono

El runtime de Mono implementa un motor JIT ( Just In Time) para la máquina virtual CIL (Common Intermediate Language), el cargador de clases, el recolector de basura, un sistema de gestión de memoria y librerías de acceso a metadatos.

Sus características son:

1. Carga y verificación de las imágenes CIL (Common Intermediate Language)
2. Ejecución de las imágenes CIL
3. Recolección de basura
4. Soporte multihilo y de E/S

5. Interacción con librerías ya existentes (P/Invoke)
6. Sistema de seguridad
7. mono: es el compilador JIT (Just in Time), el cual usa un selector de instrucción BURS.
8. mint: Es un interprete de Mono. Es un motor runtime muy fácil de portar

Mint fue originalmente desarrollado como una prueba del concepto de mono. Fue diseñado para ser fácil de depurar, fácil de estudiar, y los suficientemente comprensivo como para ser usado como una referencia para los problemas de depurado con el motor JIT. Mint es más portable que JIT, y un efecto de esto es que se puede correr Mono en diferentes arquitecturas sin mucho trabajo. Idealmente, se ha creado JIT para cada plataforma soportada, pero el intérprete es más útil cuando se quiere correr a Mono muy rápido, y correrlo bajo sistemas donde la velocidad es lo más importante.

### 3.3.1. Mono JIT

El JIT (Just in Time) de Mono traslada las instrucciones CIL a código nativo en tiempo de ejecución. El JIT compila un ensamblado entero en una sola pasada, o un método a la vez; y en la primera pasada cada método es invocado individualmente.

El JIT usa un conjunto de macros que generan código en un buffer de memoria. Mono necesita un conjunto de macros por cada arquitectura. Estos macros simplifican la generación de código, el depurado y el prototipado. La Interfaz de generación de código para la plataforma de computadora de clase x86 se encuentra en el archivo mono/arch/x86/x86-codegen.h. Los macros x86-codegen.h se originaron en Plataforma de Maquina Virtual de Java de Intel. El equipo de Mono convirtió esos macros para ser usados desde C, el lenguaje en el cual esta escrito el JIT.

La conversión del código CIL a instrucciones nativas es donde las cosas se tornan interesantes. Mono usa un selector de instrucción basado en el sistema BURS (Bottom-up Rewrite Systems), la misma tecnología usada por el compilador portable Icc ANSI C.

BURS usa una gramática que mapea un conjunto de operaciones (los nodos terminales) a elementos no terminales que se combinan con la arquitectura destino. Esta gramática esta aplicada en un programa generador de código, monoburg. Los conflictos son resueltos usando funciones asociadas con cada producción. La combinación de patrones de entrada es un árbol de operaciones. Este mapea el árbol a la arquitectura destino seleccionando los nodos que tienen un costo total mínimo asociado con ellos.

El primer paso transforma una secuencia de instrucciones CIL en un bosque de árboles. Cada árbol tiene que ser alimentado al selector de instrucción separadamente. Durante este proceso de creación de bosques y árboles, las instrucciones CIL estándar son transformadas en códigos que son mejor combinadas

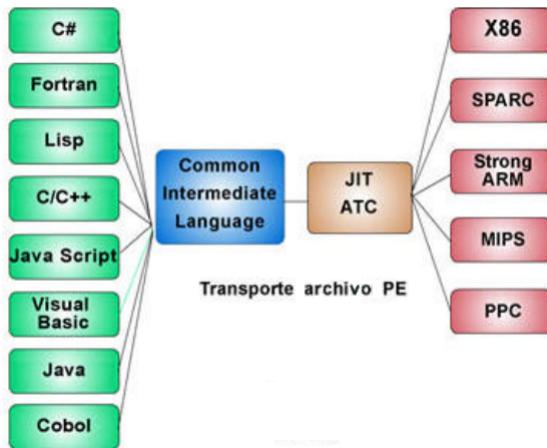


Figura 3.3: Compilación de JIT

con el selector de instrucción. Para generar el código, un número de pasadas son realizadas en los árboles de nodos. El primer pase etiqueta todos los nodos y encuentra el árbol más económico, y en el segundo pase realiza la asignación del registro. La etapa final emite el código x86. En esta escritura, el motor JIT (Just in Time) soporta la mayoría de las características no orientadas a objetos de la máquina virtual. Para cuando se las lea, las características orientadas a objetos deberían ser implementadas.

En la Figura 3.3 se observa la compilación de JIT en cada uno de sus pasos.

### 3.3.2. Multi-threading

Multi-threading es la capacidad de ejecutar simultáneamente diferentes hilos. La programación más tradicional básica es secuencial: después de una instrucción sabemos que se ejecutará la siguiente. Pero si trabajamos con hilos (threads) podemos hacer que se ejecuten dos o más flujos de instrucciones simultáneamente, permitiendo por ejemplo, seguir realizando cálculos mientras se espera la entrada por teclado del usuario de algún dato.

### 3.3.3. PInvoke

PInvoke (Platform Invoke) es el mecanismo que se usa para envolver las llamadas de los API's de Unix tan bien como si se estuviera hablando a las librerías del sistema. Con PInvoke podremos hacer llamadas a librerías, clases o programas en general antiguos, creados en lenguajes diferentes como C o C++ que están fuera de la plataforma .Net. De esta forma se pueden construir librerías intermedias para lenguajes .Net que permita el acceso a funcionalidades implementadas en librerías antiguas.

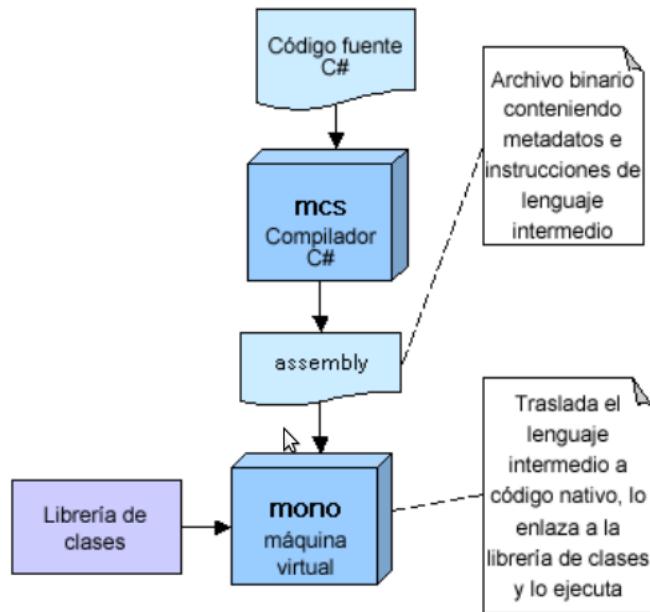


Figura 3.4: Compilador C# de Mono

En este caso se encuentra GTK#, el cual hace uso de PInvoke para acceder a las librerías gráficas GTK originales y así proporcionar a lenguajes como C# la posibilidad de trabajar gráficamente usando GTK#.

### 3.4. Compilador C# de Mono

MCS es el compilador C# de Mono. Algo muy interesante es que está escrito en el mismo C# y es considerado altamente portable. El compilador C# de Mono es considerado completo hasta este punto y relativamente maduro. MCS esta habilitado para compilarse por si mismo y muchos programas C#. Se lo utiliza para compilar Mono, aproximadamente un millón de líneas del código de C#. Hay un par de áreas que no están cubiertas por el compilador Mono, pero son muy pocas en este punto, se trata de atributos de seguridad. MCS está escrito en C# y usa todo el potencial de los API's de .Net. MCS corre en Linux (con el runtime de Mono) y en Windows (con el runtime de marco de referencia .Net) MCS no compila código nativo, pero si un tipo de bytecode, que puede ser procesado por el tiempo de ejecución de Mono, como nos muestra la Figura 3.4.

Cuando se reporta un problema, se trata de proveer un test pequeño que puede mostrar el error, este test se lo incluye como una parte de la prueba de regresión de Mono C#. Si este fallo es un error o un aviso de advertencia que

no tiene una bandera en el test, entonces se crea un programa simple llamado csXXXX.cs, donde XXXX es el número de código que es usado por el compilador C# que ilustra el problema. El compilador tiene un número de fases:

1. Analizador Léxico: analizador léxico codificado a mano, que provee tokens al parser.
2. El Parser (intérprete): el parser es implementado usando Jay (se utiliza para portear a Java, y ahora a C#). El parser hace un trabajo mínimo y un chequeo sintáctico, y únicamente construye un árbol interpretado. Cada elemento del lenguaje obtiene sus propias clases. La convención de código usa un nombre en mayúscula para el elemento del lenguaje. Así una clase C# y su información asociada es mantenida en una clase “Class”, una estructura en una clase “Struct” y así sucesivamente. Las declaraciones se derivan de la clase “Statement”, y las expresiones de la clase “Expr”.
3. Resolución de la clase padre: antes de la generación de código actual, se necesita resolver las interfaces las clases padre para la interfaz, las clases y las definiciones de estructura.
4. Análisis semántico
5. Generación de código: La generación de código se la realiza a través del API System.Reflection.Emit

### 3.5. Biblioteca de clases externas

Además de la implementación del estándar .Net el marco de referencia de mono frese otras herramientas que aumentan la funcionalidad del mismo, a continuación se listan algunas de estas:

IKVM nos ofrece una máquina virtual java que corre sobre mono de forma que podemos ejecutar bytecode desde mono, y quizás lo más interesante: un compilador de bytecode a CIL.

Monodoc es el visualizador de documentación de Mono y ofrece dos interfaces, vía web (servidor XSP) o vía aplicación gráfica (GTK#).

NAnt gestor de compilación de código de filosofía similar al Ant de Java, es decir, pierde algo de flexibilidad comparado con make y sus Makefiles pero gana en portabilidad ya que en los archivos de configuración no se indican comandos nativos del sistema operativo, sino etiquetas XML que luego serán ejecutadas por NAnt según el sistema donde se encuentre.

GTK# + Glade recubrimiento de las conocidas bibliotecas gráficas GTK para poder ser usadas desde la plataforma Mono. Con Glade es posible diseñar interfaces gráficas de forma completamente visual y guardarlas en archivos XML, después desde nuestro programa utilizando las bibliotecas de Glade# podemos cargar dicha información para que sea visualizada por pantalla con GTK# y interactuar con ella, por ejemplo asociando métodos a los eventos.

## Capítulo 4

# Herramientas integrales de desarrollo

Actualmente existen infinidad de IDE's denominados Herramienta Rápida de Desarrollo (RAD) que nos permiten desarrollar nuestras aplicaciones de una manera mas amigable, el surgimiento de estos IDE's se debe a la gran demanda de aplicaciones en periodos de tiempo cortos estas herramientas permiten reducir el tiempo para crear la aplicación considerablemente, sin embargo algunas de estos IDE's son lentos a la hora de compilar y ejecutar las aplicaciones lo cual no es de tanta importancia ya que se llevaría mayor tiempo si se hiciera la aplicación de manera nativa.

A continuación se mencionan algunos de los IDE's mas conocidos de la actualidad.

### 4.1. NetBeans

Netbeans [6]es un IDE open source de distribución gratuita, que cuenta con un soporte técnico que muy pocos programas por paga lo igualan. Cuenta, además, con el respaldo de una comunidad que aporta “plugins”, lo que permite personalizar el IDE hasta que cumpla nuestros Estándares personales.

Lo más interesante de este IDE es la integración de los diferentes módulos que pueden ser activados para permitir la creación de programas en lenguajes como Java, J2ME, PHP, C++, Python, Ruby, JSP, entre otros. Haciendo un poco de historia, podemos encontrar que Netbeans surgió como un proyecto llamado Xelfi , en 1996 creado por Roman Stank, un ciudadano Checo. En 1999 Sun Microsystems adquirió el proyecto continuando con el nombre de Netbeans y haciéndolo open source. Desde 2004, la plataforma es actualizada más de una vez por año. En el momento de escribir este artículo, la versión 6.9 esta siendo evaluada para pasar a ser la 7.0, por las importantes mejoras que incorpora.

En la práctica, las ventajas que podemos tener con Netbeans son muchas, trataremos de enumerar algunas con una breve descripción:

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

- Auto-completa el código que escribimos: Ante la falta de inicialización de algún argumento, nos sugiere la declaración automática del mismo; también, nos propone las características disponibles para los elementos, cuando intentamos acceder a estas mediante el punto después de la variable o argumento.
- Función de Importar Clases: Si hacemos uso de una clase para la cual no hemos hecho previamente la declaración de importación a nuestro código, nos permite con un simple CRTL + SHIFT+I, resolver todos esos errores.
- Diseño Visual: Se pueden crear formularios y ventanas de forma visual, en diferentes plataformas que van desde J2ME con formularios para aplicaciones móviles así como diseño de canvas para juegos con el “Visual Game Desing”. Para Java SE, permite utilizar toda la librería Swing en la creación visual.
- Integración de Servidores: Como podemos crear diferentes aplicaciones al trabajar en diferentes lenguajes, Netbeans trae en su plataforma servidores Web y de aplicaciones (su instalación es opcional), por lo que si queremos crear un servicio y probarlo desde una aplicación en el desktop, podemos arrancar localmente el servidor de nuestra preferencia desde Netbeans, y una vez probemos la aplicación, ésta será desplegada automáticamente por el servidor, evitando la compilación y la carga posterior al servidor, manualmente.
- Web Services: Por defecto, Netbeans trae algunos Web Services de los principales sitios de Internet (Google, Facebook, Yahoo, Flickr, Vicious, entre otros) pre-conFigurados, con los cuales podemos hacer pruebas e incluso aplicaciones. Además de que podemos agregar alguno de nuestra preferencia o creación. De spués de agregados, sólo tenemos que hacer un “Drag and Drop (Agarrar y Soltar)” en nuestra aplicación, para lograr una referencia a este Web Service.
- Emuladores: Para la creación en plataformas como J2ME, la cual tiene que cumplir con ciertas características muy específicas para cada perfil, nos permite la integración de los SDK de los fabricantes y sus emuladores, así podemos asegurarnos de cumplir hasta el mínimo requisito de cada conFiguración. Además de todo lo mencionado anteriormente, en [www.Netbeans.org](http://www.Netbeans.org) podrás encontrar ejemplos y tutoriales de cómo sacar el mayor provecho de la aplicación; así también técnicas para mejorar el desempeño de nuestros programas. Le invito visite el website de Netbeans si tiene algún interés en programación o por simple curiosidad como yo la tuve hace unos años; ahora, hago dinero con este programa y lo que he aprendido.

En la Figura 4.1 se observa el IDE NetBeans en su versión 7.0.1, al crear un proyecto Java.

### 4.2. Eclipse

La plataforma Eclipse [3] consiste en un Entorno de Desarrollo Integrado, abierto y extensible. Como elementos básicos, este IDE cuenta con en un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

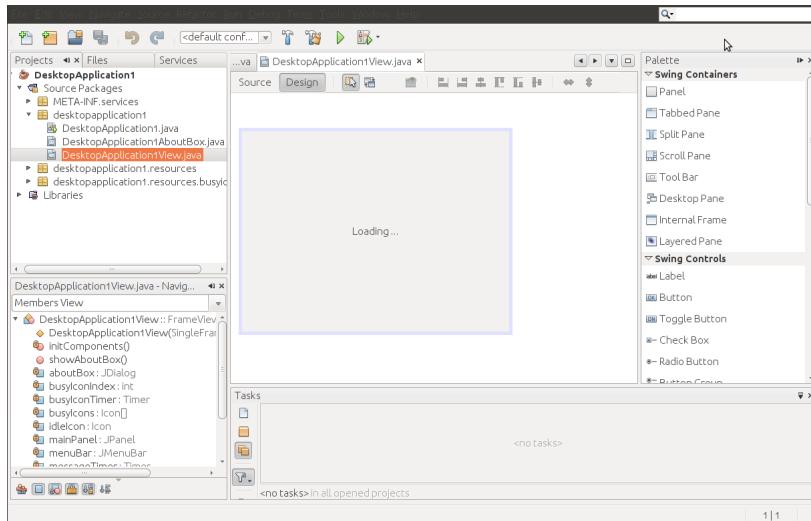


Figura 4.1: IDE NetBeans v7.0.1

(plugins) para extender la funcionalidad.

El término Eclipse además identifica a la comunidad de software libre para el desarrollo de la plataforma Eclipse. Este trabajo se divide en proyectos que tienen el objetivo de proporcionar una plataforma robusta, escalable y de calidad para el desarrollo de software con el IDE Eclipse. Este trabajo está coordinado por la Fundación Eclipse, que es una organización sin ánimo de lucro creada la promoción y evolución de la plataforma Eclipse dando soporte tanto a la comunidad como al ecosistema Eclipse.

Gran parte de la programación de Eclipse fue realizada por IBM antes de que se creara el proyecto Eclipse como tal. El antecesor de Eclipse fue VisualAge y se construyó usando Smalltalk en un entorno de desarrollo llamado Envy. Con la aparición de Java en la década de los 90, IBM desarrolló una maquina virtual válida tanto para Smalltalk y Java. La rápida expansión de Java y sus ventajas con miras a una Internet en plena expansión obligaron a IBM a plantearse el abandono de esta maquina virtual dual y la construcción de una nueva plataforma basada en Java desde el principio. El producto final resultante fue Eclipse, que ya había costado unos 40 millones de dólares a IBM en el año 2001.

A finales de 2001 IBM, junto a Borland, crearon la fundación sin ánimo de lucro Eclipse, abriendose así al mundo de código abierto. A este consorcio se han unido progresivamente importantes empresas del desarrollo de software a nivel mundial: Oracle, Rational Software, Red Hat, SuSe, HP, Serena, Ericsson, Novell, entre otras. Hay dos ausencias significativas: Microsoft y Sun Microsystems. Microsoft ha sido excluida por su posición de monopolio del mercado, y Sun Microsystem cuenta con su propio IDE y principal competencia de Eclipse: NetBeans. De hecho, el nombre de Eclipse fue elegido porque el objetivo era crear un IDE capaz de "eclipsar a Visual Studio" (Microsoft) así como "eclipsar

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

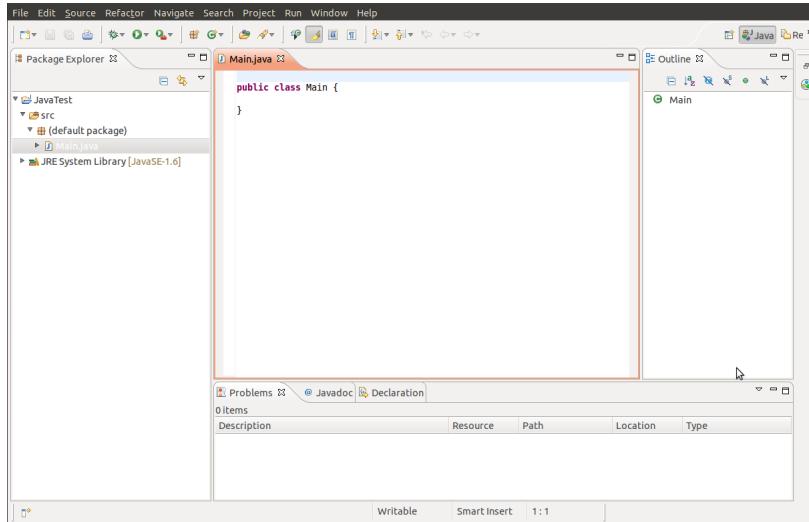


Figura 4.2: IDE Eclipse v3.7.2

el NetBeans" (Sun Microsystem).

La última versión estable de Eclipse se encuentra disponible para los sistemas operativos Windows, Linux, Solaris, AIX, HP-UX y Mac OSX. Todas las versiones de Eclipse necesitan tener instalado en el sistema una máquina virtual Java (JVM), preferiblemente JRE (Java Runtime Environment) o JDK (Java Developer Kit) de Sun, que a principios de 2007 no son libres (aunque hay un anuncio por parte de Sun de que lo serán).

En la Figura 4.2 se muestra el IDE eclipse versión 3.7.2, este es el aspecto de este entorno al crear un proyecto Java.

### 4.3. Visual Studio

Visual Studio .Net [4]es la Herramienta Rápida de Desarrollo (RAD) de Microsoft para la siguiente generación de Internet que son los Servicios Web XML. Esta herramienta permite la creación de aplicaciones usando marco de referencia .Net, es decir usando el CLR, la Librería de Clases, ADO .Net, ASP .Net, etc. Propio para los sistemas operativos Microsoft Windows.

Es un software que brinda las herramientas necesarias para crear, distribuir, administrar y dar mantenimiento a aplicaciones Web distribuidas que usan Servicios Web XML, todo esto con una gran facilidad, rapidez y bajo costo.

Se puede crear aplicaciones Web directamente usando el marco de referencia .Net y algún programa editor, por ejemplo el Bloc de Notas, pero el tiempo que llevaría el desarrollo no justificaría el ahorro de costos, en cambio, si se utiliza una herramienta como Visual Studio .Net el tiempo de desarrollo se reduciría enormemente.

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

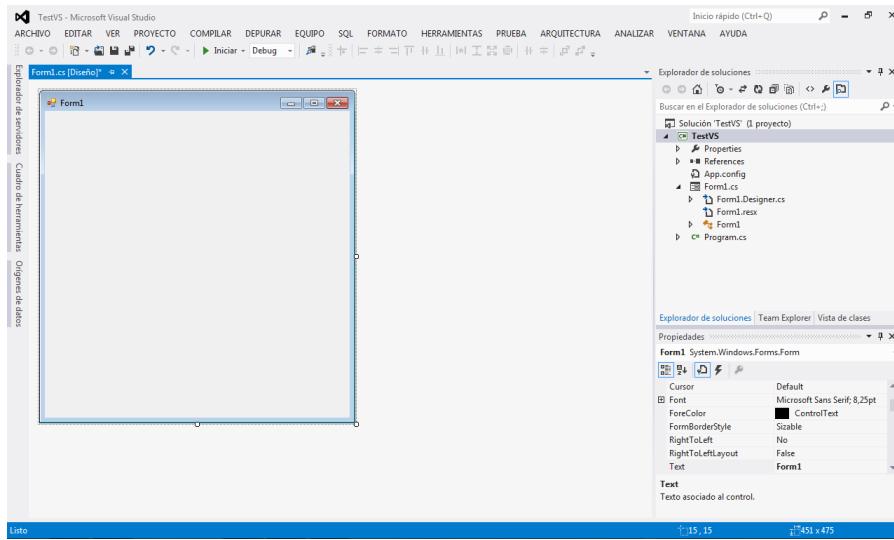


Figura 4.3: IDE Visual Studio 2012

Visual Studio .Net permite también la integración y el uso cruzado de lenguajes de programación: Visual Basic .Net, Visual C# .Net, Visual C++ .Net y JScript .Net.

En la Figura 4.3 se muestra el IDE visual Studio 2012 que utiliza marco de referencia .Net para el desarrollo de aplicaciones.

### 4.4. WinDev

WinDev [7] es un IDE poderoso creado por la sociedad francesa PC SOFT, que permite crear aplicaciones basadas en un motor de ejecución (marco de referencia).

Herramienta concebida para desarrollar rápidamente aplicaciones, principalmente orientadas a datos. El lenguaje de programación incluido en el útil, el W-Language, un lenguaje de 4º generación (L4G). Es un lenguaje simplificado y flexible que permite obtener resultados muy rápidamente.

Posee un editor de interfaz gráfica que permite crear interfaces de usuario gráficas por medio de drag/drop. Permite también escoger un modelo de manual de estándares gráfico a partir de una lista que se propone o crear uno nuevo. La herramienta de interfaz permite definir interactivamente varios aspectos sobre la entrada de datos: tamaño, máscara, formato automático, entrada obligatoria. Esto limita también el número de funciones a programar. También maneja herencia y sobrecarga. El código se pre-compila e interpreta en la ejecución por el marco de referencia, lo cual permite independencia del archivo ejecutable con respecto al sistema operativo. WinDev fue inicialmente previsto para Windows. WinDev permite generar aplicaciones en Java así como aplicaciones estándares

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

o aplicaciones para la plataforma .Net o aplicaciones Linux nativas (desde la versión 16) con uso de la biblioteca Qt, pero no es totalmente compatible. Sus hermanos WebDev y WinDev Mobile permiten utilizar el mismo lenguaje de programación y los mismos conceptos (análisis, ventanas, reportes, componentes, clases...), para la generación de sitios Web y de aplicaciones para Pocket PC, Smartphones y terminales industriales.

WINDEV® 17 es la mejor solución para el desarrollo de aplicaciones, ya sean en una arquitectura distribuida o con entornos cliente/servidor, red local, cliente enriquecido. Desarrollando 10 veces más rápido con un entorno totalmente integrado, su equipo de desarrollo puede alcanzar los objetivos que se le ha dado: se respetan las fechas límite y los presupuestos. Le permite crear fácilmente nuevas aplicaciones compartidas (a través de Windows, el Internet, la Intranet) que aumentan la receptividad y la rentabilidad de las compañías.

Usa Web Services fácilmente: .Net, J2EE. WINDEV® 17 automáticamente genera reportes PDF, con códigos de barra, fondos, etc.

Herramienta CASE completamente abierta a:

- Lenguajes de 3<sup>a</sup> y 4<sup>a</sup> generación: Java, C++, C#, VB, Cobol, Fortran, etc.
- Bases de datos: MySQL, SQL Server, DB2, AS/400, Access, etc.

### 4.5. MonoDevelop

MonoDevelop [5] es un entorno de desarrollo diseñado principalmente para C # y otros lenguajes .Net. MonoDevelop permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y Web ASP.Net en Linux, Windows y Mac OSX. MonoDevelop hace que sea fácil para los desarrolladores a puerto. NET creadas con Visual Studio para Linux y mantener una base de código único para todas las plataformas.

#### 4.5.1. Características

- Multi-plataforma: Compatible con Linux, Windows y Mac OSX.
- Edición de texto avanzada: Código de finalización de soporte para C # 3, plantillas de código, plegado de código.
- Banco de trabajo conFigurable: Son totalmente conFigurables, diseños de ventana, atajos de teclado definidos por el usuario, herramientas externas.
- Soporte para múltiples lenguajes: C #, Visual Basic.Net, C / C ++, etc.
- Depurador integrado: Para depuración de Mono y aplicaciones nativas
- GTK # Visual Designer: Cree fácilmente aplicaciones GTK #
- ASP.Net: Crear proyectos web con soporte completo, completado de código y prueba en XSP, el servidor web de Mono.

## CAPÍTULO 4. HERRAMIENTAS INTEGRALES DE DESARROLLO

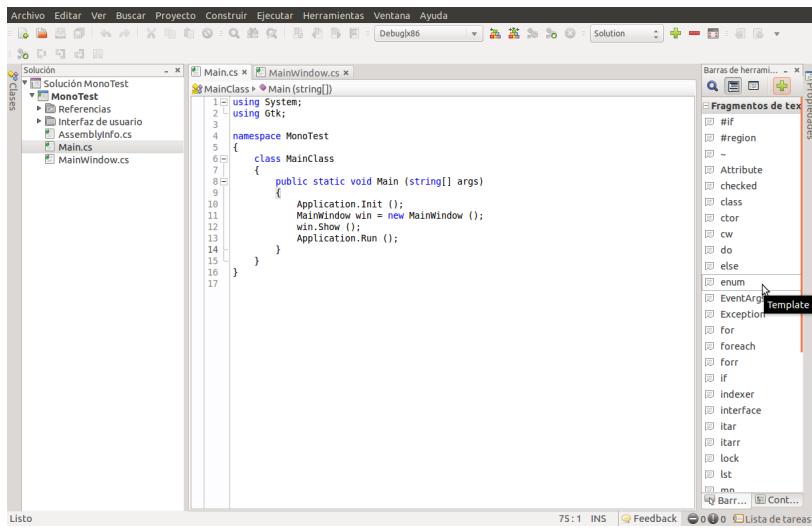


Figura 4.4: MonoDevelop v2.8.6.3

- Otras herramientas: Control de código fuente, integración de makefile, pruebas unitarias, packaging and deployment, localización.

En la Figura 4.4 se muestra el IDE MonoDevelop que es utilizado para el desarrollo de aplicaciones con la tecnología Mono.

## Capítulo 5

# Trabajo relacionado

En [16] trata de un estudio comparativo y análisis crítico de los diferentes entornos de desarrollo integrado de C, C + +, y Java para que una lista completa de las especificaciones puede extraerse para el desarrollo de una IDE futuro régimen de todo incluido y óptimo. Un entorno de desarrollo integrado (IDE), también conocido como un entorno de diseño integrado o un entorno de depuración integrada, es una aplicación de software que proporciona servicios integrales para los usuarios de computadoras para saber, desarrollo de software, entrada del programa, edición, compilación, depuración, etc IDE's Varios son estudiados y comparados en función de sus características, y se analizan críticamente. Los resultados muestran que los IDEs son más populares que suelen ofrecer un mejor soporte para los usuarios, mientras que los más complejos, que no son tan populares, ofrecen características sólo unos pocos que el otro IDE IDE existente, mientras que no es totalmente completa y óptima.

En el [14] presenta una síntesis de los resultados obtenidos en la comparación de la plataforma J2EE frente a la plataforma .Net en aspectos concernientes a los Servicios Web, mostrando las principales fortalezas y debilidades que se pueden obtener al escoger entre las dos plataformas.

El trabajo [15] de tesis tiene el objeto conocer a través de un análisis comparativo entre las arquitecturas Asp.Net Tradicional y Asp.Net MVC en Visual Studio 2010, cual es el más óptimo para el desarrollo de aplicaciones web.

De acuerdo al análisis comparativo, mediante los indicadores de comparación Interoperabilidad, Portabilidad, Escalabilidad, Acceso a Datos y Programación, además se usaron para cada indicador varios parámetros y para cada parámetro varios índices, de donde se determinó un 90.54 % para la arquitectura Asp.Net Tradicional (Formularios Web) con Visual Studio .Net 2010 y 83,52 % la arquitectura Asp.Net MVC con Visual Studio .Net 2010, respectivamente; y por ende se concluyó que la arquitectura Asp.Net Tradicional (Formularios Web) con Visual Studio .Net 2010 es la mejor opción y es la que más prestaciones de desarrollo y optimización ofrece por ahora, ya que cada día se están desarrollando metodologías que a posteriori es probable que cambien.

Por ahora aún Asp.Net Web Forms, se puede considerar como el más re-

## CAPÍTULO 5. TRABAJO RELACIONADO

comendado para desarrollar aplicaciones web, sobre todo porque permite crear aplicaciones de manera ágil, optimizando el tiempo, por lo tanto también se optimiza los recursos de manera general, el gran inconveniente con MVC es que la curva de aprendizaje es muy elevado, situación que debería ser tomado en cuenta a la hora de elegir una arquitectura para el desarrollo de sistemas o aplicaciones.

En este trabajo [11] Microsoft ha publicado .NET, una norma depende de la plataforma para el lenguaje de programación C#. Patrocinado por Ximian / Novell, Mono, la plataforma de desarrollo de código abierto basado en el marco de referencia .Net, se ha desarrollado para ser una versión independiente de la plataforma de el entorno de programación C#. Mientras .Net es dependiente de la plataforma, Mono permite a los desarrolladores construir aplicaciones Linux y multiplataforma. Mono se basa en los estándares de ECMA para C#.

Este artículo examina estos dos entornos de programación, con el objetivo de evaluar las características de funcionamiento de cada uno. La prueba se realiza con diferentes algoritmos. Además, se evalúan las ventajas y desventajas asociadas con el uso de una plataforma cruzada frente a una implementación dependiente de la plataforma.

En [10] menciona que con el auge de las plataformas móviles y la necesidad de estar siempre conectado, se hace necesario el desarrollo de servicios web multiplataforma que cumplan con la demanda de los usuarios.

.Net es un marco de referencia de Microsoft en el que destaca la transparencia de redes, independientemente de la plataforma de hardware. Su principal característica es que ofrece una manera rápida y económica de desarrollar aplicaciones permitiendo una integración más rápida y eficaz, y simplificando el acceso a información desde cualquier tipo de dispositivo.

En este proyecto se ha realizado el estudio de esta tecnología, proponiendo una aplicación chat para la comunicación tanto en formato texto como en vídeo llamada a través de la conocida aplicación Skype, y desde cualquier dispositivo con conexión a internet, ya sea pc, Tablet o dispositivos móviles.

En el artículo [9] se presentan el proyecto SCRIBO (Semi-automatic and Collaborative Retrieval of Information Based on Ontologies) y se describe cómo aprovechar el marco UIMA con el fin de integrar las herramientas existentes en una arquitectura general. El documento se centra en la forma en que han tendido un puente de plataformas Java y .Net (usando el marco de referencia de Mono), describiendo los problemas y una solución eficaz para hacer posible la interoperabilidad UIMA.

## Capítulo 6

# Desarrollo de aplicaciones utilizando software propietario

### 6.1. Creación de un nuevo proyecto

Para la creación de un nuevo proyecto en visual studio 2012 daremos click sobre “Nuevo proyecto...” o a través de la barra de herramientas “ARCHIVO->Nuevo->Proyecto” en la Figura 6.1 se muestra la pantalla inicial de visual studio 2012.

El segundo paso es seleccionar el tipo de aplicación que en este caso es una un aplicación de escritorio para ello seleccionamos Visual C# y elegimos aplicación windows form, como se muestra en la Figura 6.2. Al dar aceptar se creara nuestro proyecto, la Figura 6.3 muestra el resultado de la creación de un proyecto windows form.

### 6.2. Creación de la base de datos

A diferencia de MonoDevelop visual studio 2012 ofrece una herramienta dentro de este para administrar una base de datos. Para la creación de la base de datos nos dirigimos con el puntero del ratón a el apartada *Explorador de servidores* que se encuentra en el índice superior izquierdo de la pantalla, para agregar una nueva conexión damos click sobre el icono de un enchufe o dando click derecho sobre Conexiones de datos->nueva Conexión, nos abrirá una nueva venta que pedirá el origen de los datos y el nombre de nuestro archivo que en este caso sera ITM la Figura 6.4 muestra cada uno de los elementos antes mencionados, posterior a esto solo falta aceptar y se creara nuestra base de datos.

Al tener nuestro archivo *ITM.mdf* creado daremos click derecho *agregar*

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

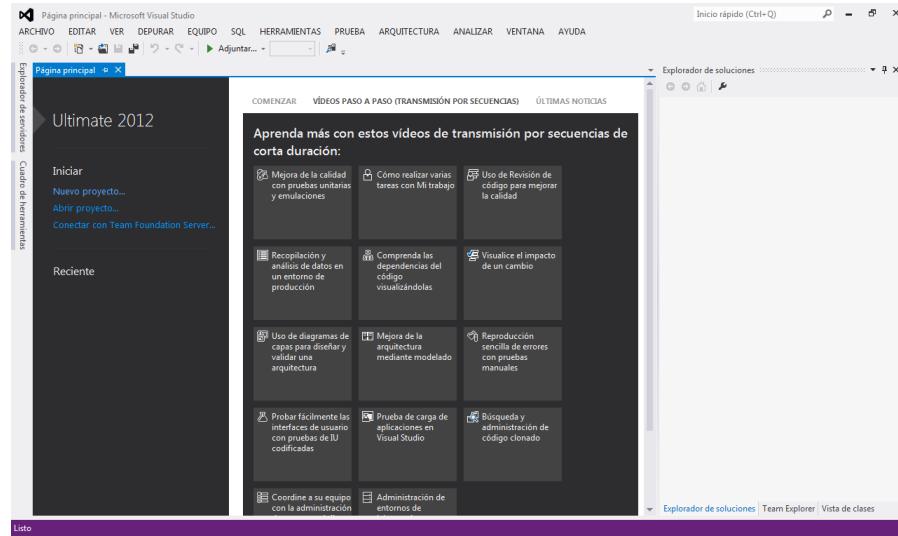


Figura 6.1: Pantalla inicial de visual studio 2012

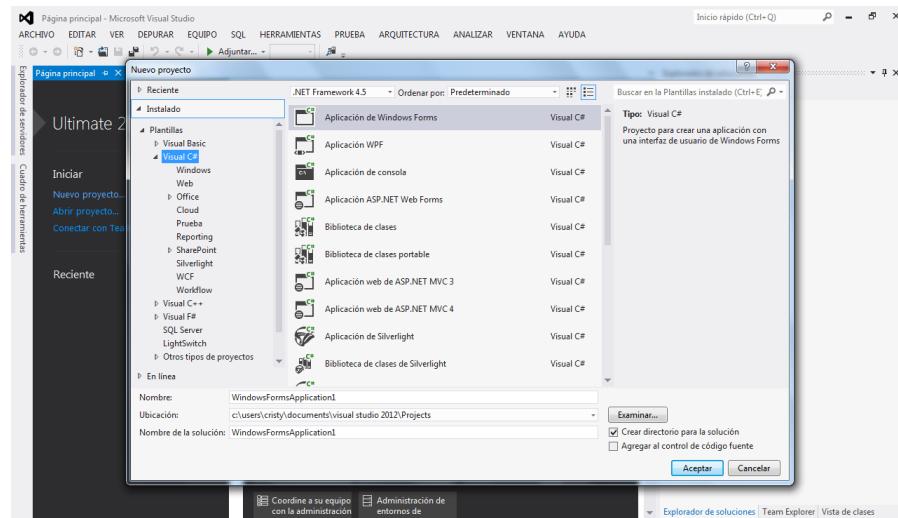


Figura 6.2: Creando aplicación windows form

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

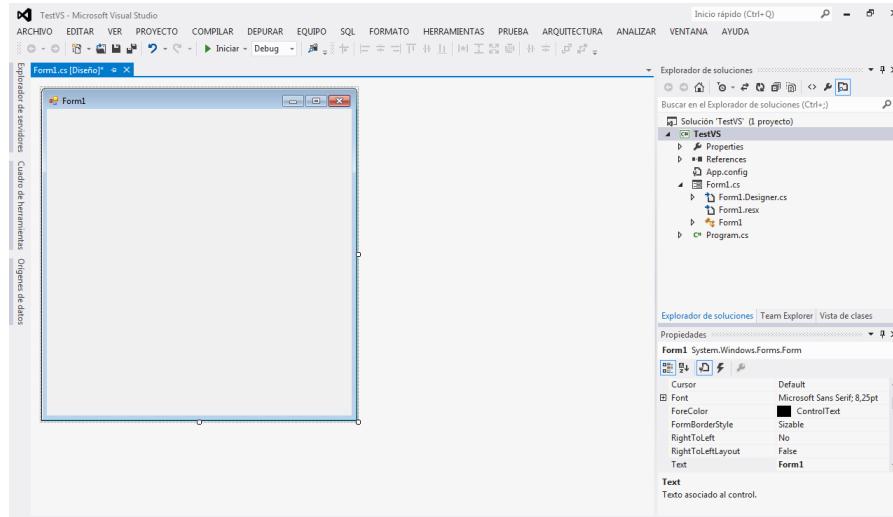


Figura 6.3: Aplicación windows form

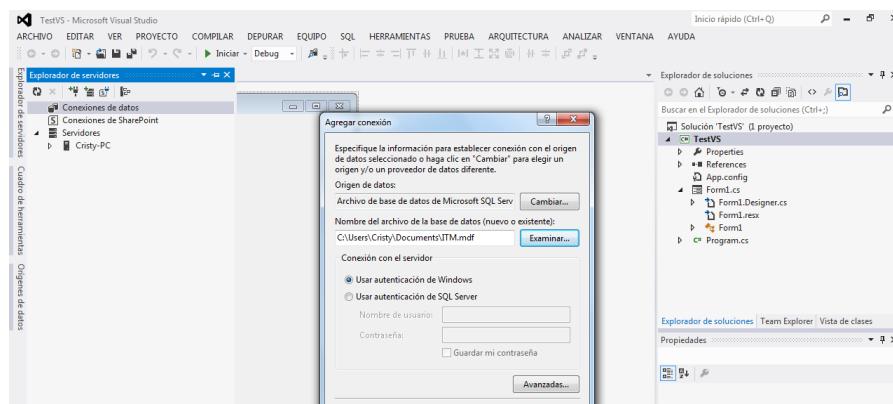


Figura 6.4: Nombrando base de datos

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

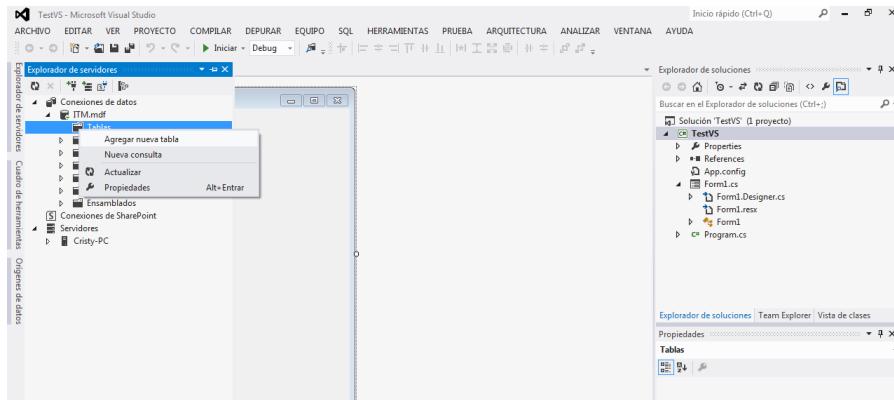


Figura 6.5: Agregar nueva tabla a la base de datos

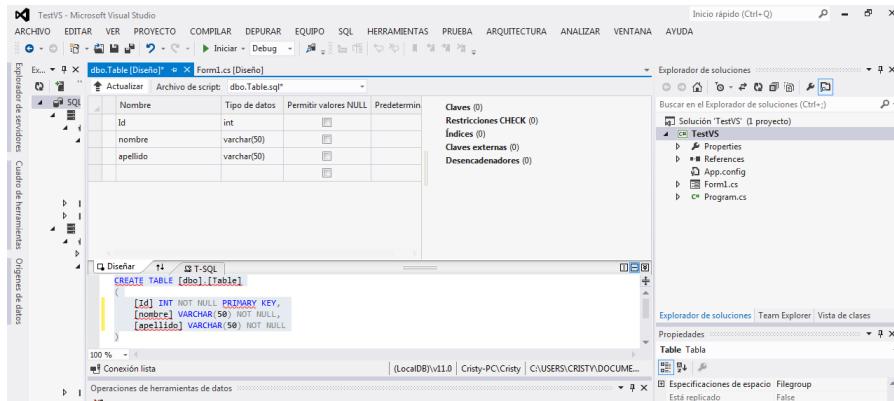


Figura 6.6: Diseño de la tabla usuarios

nueva tabla tal como se muestra en la Figura 6.5. Al hacer esto se mostrara en pantalla el diseño de la tabla a crear la cual tendrá que quedar tal como se muestra en la Figura 6.6. Teniendo nuestra tala creada daremos click derecho sobre esta y seleccionamos mostrar datos de tabla, en primera instancia aparecerá bacia, la Figura 6.7 muestra el formato que se visualiza pero ya con algunos elementos.

### 6.3. Creación de la ventana principal

A diferencia de MonoDevelop que fue necesario dividir la ventana para la colocación de los componentes, en visual studio nos permite arrastrar los componentes directamente sobre nuestra ventana y permite definir su tamaño tan solo con sostener un click al borde y darle la dimensión requerida. La Figura 6.8 muestra como agregar un *DataGridView* área que mostrara la información de la

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

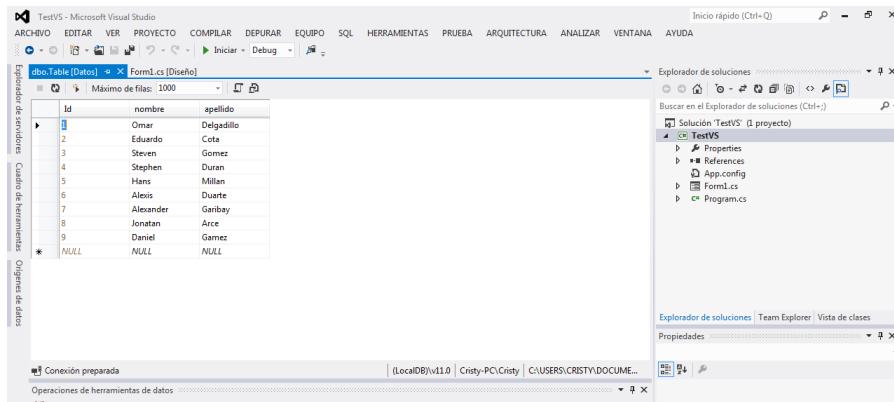


Figura 6.7: Agregando elementos la tabla

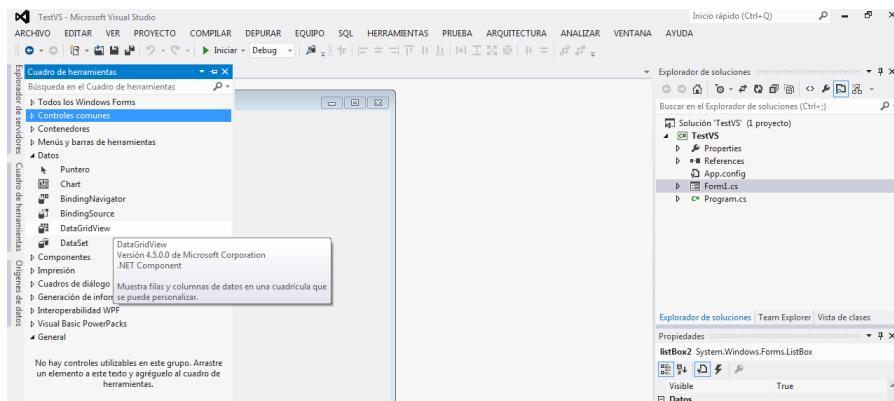


Figura 6.8: Creando DataGridView para mostrar base de datos

base de datos. La Figura 6.9 muestra que en primer instancia el DataGridView esta vacío no tiene ningún entrada de datos, para agregar la tabla tendremos que dar click sobre *Agregar origen de datos del proyecto...*, nos abrirá una nueva ventana el la que tendremos que seguir una serie de pasos el primero sera seleccionar el tipo de origen de datos que en este caso es una base de datos, al dar siguiente nos pedirá elegir un modelo de bases de datos que en este caso es conjuntos de datos. Elegiremos la conexión de datos ITM.mdf como se muestra en la Figura 6.10, guardamos la cadena conexión en el archivo de configuración de la aplicación. Finalmente seleccionaremos la tabla y datos que requerimos mostrar de ella, la Figura 6.11 muestra que tabla y datos elegimos para esta aplicación.

Teniendo el área de datos lista solo falta agregar algunos componentes para realizar las altas y las bajas de los usuarios, primero agregamos la etiqueta la Figura 6.12 muestra como queda nuestra ventana al agregar “label1”, seguido a

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

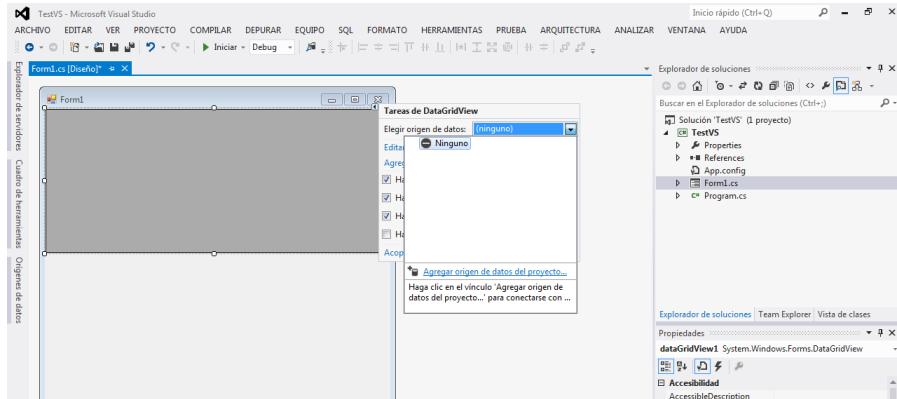


Figura 6.9: Selección del origen de los datos

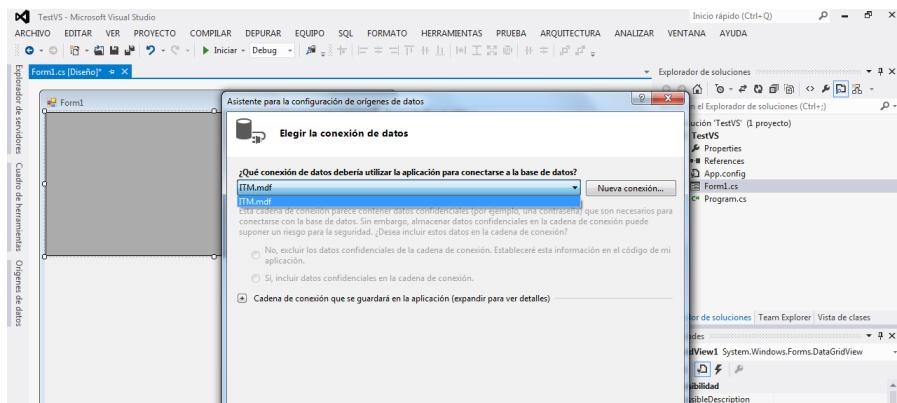


Figura 6.10: Elegir la conexión de datos

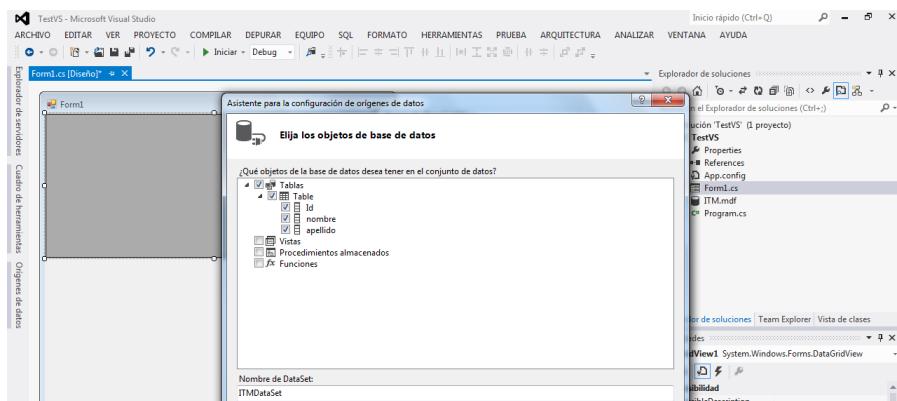


Figura 6.11: Selección de tabla y datos a mostrar

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

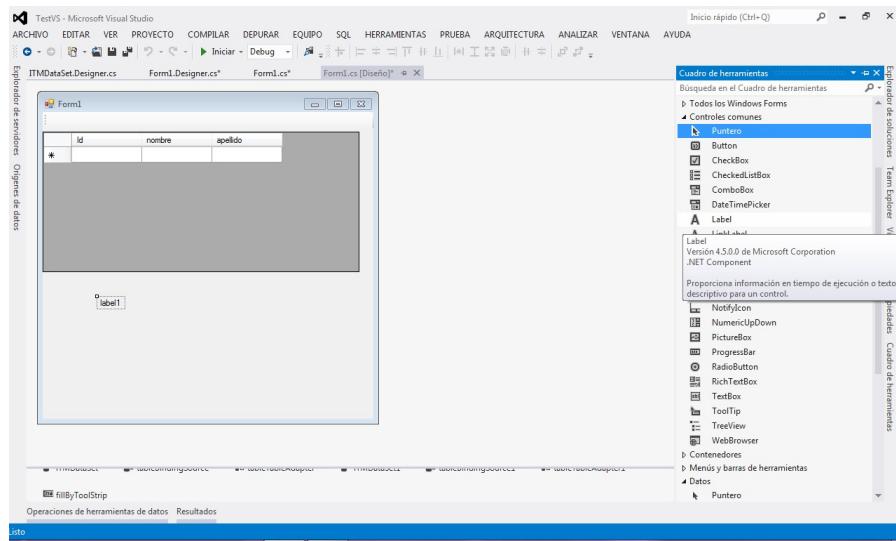


Figura 6.12: Agregando label1

esto agregaremos una entrada de texto el componente requerido es un TextBox y nuestra ventana quedara como se muestra en la Figura 6.13. Posterior a esto agregamos Label2 y TextBox2 y un botón y nuestra ventana queda como se muestra en la Figura 6.14. Agregamos label3, TextBox3 y button2 y nuestra ventana quedara tal como se muestra en la Figura 6.15. Finalmente cambiamos los nombres de nuestras etiquetas y la ventana finalmente queda como se muestra en la Figura 6.16.

### 6.4. Agregar funcionalidad a los botones

En este momento la aplicación solo mostrara los elementos de la base de datos pero no podremos realizar ningün alta o baja ya que los botones aun no tienen funcionalidad. Para agregar la funcionalidad a nuestro botonesaremos lo siguiente:

1. Para el primer botón “Agregar”, estado en modo diseño daremos doble click y este automáticamente nos creara el siguiente código :

```
private void button1_Click(object sender, EventArgs e)
{
}
```

A diferencia de MonoDevelop que teníamos que realizar la conexión y la

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

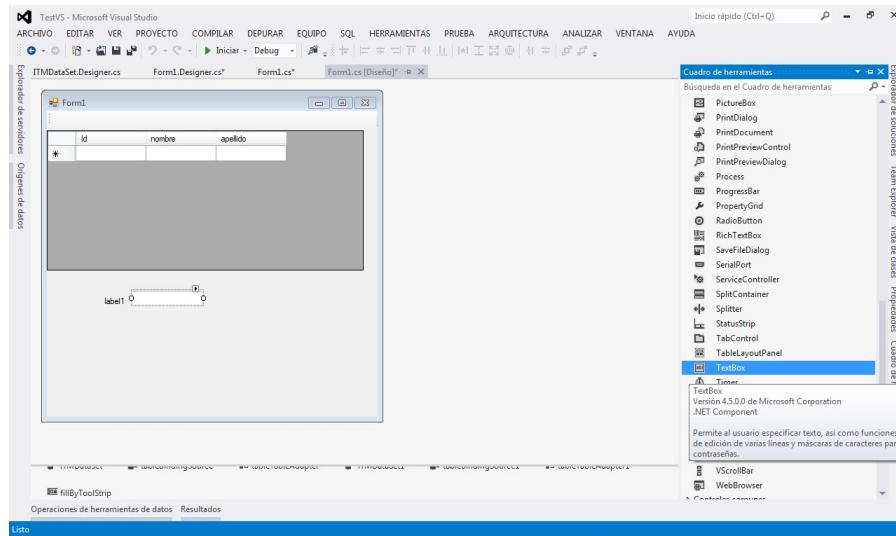


Figura 6.13: Agregando TextBox1

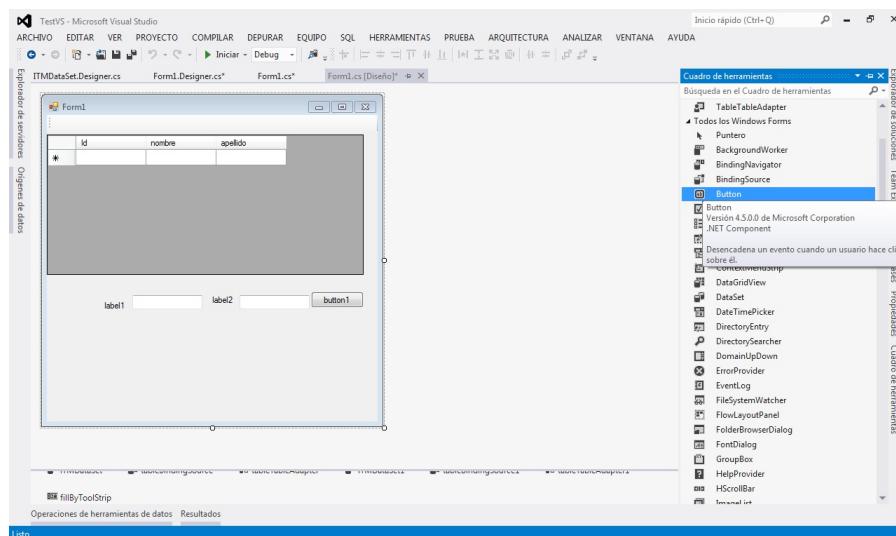


Figura 6.14: Agregando button1

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

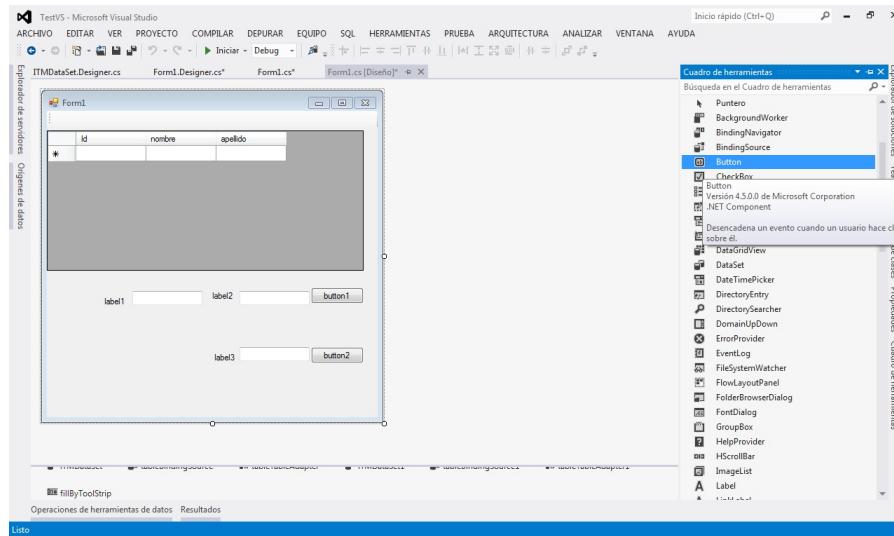


Figura 6.15: Agregando label3, TextBox3 y button2

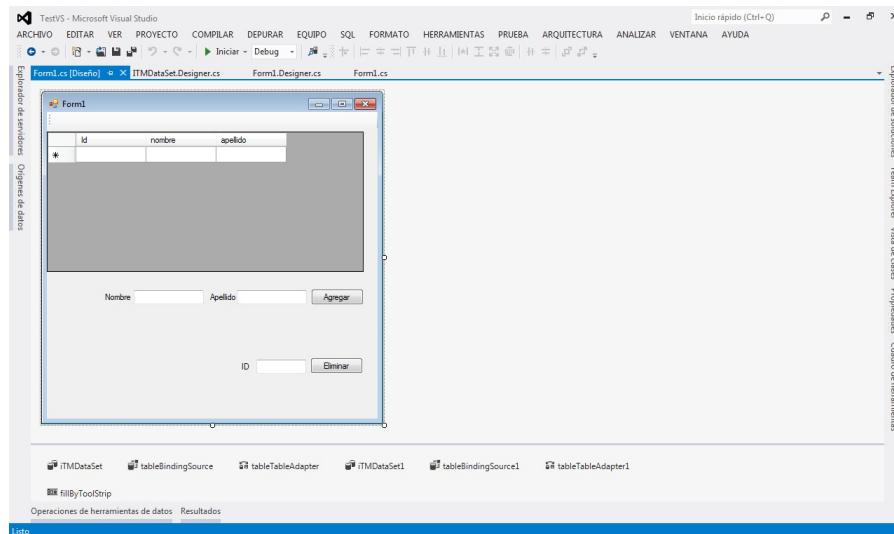


Figura 6.16: Ventana terminada

consulta manualmente en visual studio nos genera automáticamente un objeto y en este también se generan los métodos necesarios para realizar altas bajas modificaciones y consultas, por consecuente el código del botón queda de la siguiente manera:

```
private void button1_Click(object sender, EventArgs e)
{
    this.tableTableAdapter.Insert(this.textBoxNombre.Text,
        this.textBoxApellido.Text);
    this.tableTableAdapter.Fill(this.iTMDataSet.Table);
    this.textBoxNombre.Text = "";
    this.textBoxApellido.Text = "";
}
```

2. Para el segundo botón “Eliminar”, estado en modo diseño daremos doble click sobre este y automáticamente nos creara el siguiente código :

```
private void button2_Click(object sender, EventArgs e)
{
}
```

En este caso es necesario cambia la entrada de texto a un valor entero, el código siguiente muestra como quedaría terminada la funcionalidad de nuestro botón:

```
private void button2_Click(object sender, EventArgs e)
{
    int id = Convert.ToInt16(this.textBoxID.Text);
    this.tableTableAdapter1.Delete(id);
    this.tableTableAdapter.Fill(this.iTMDataSet.Table);
    this.textBox.Text = "";
}
```

## 6.5. Aplicación en ejecución

Con todos los cambios hechos a el código nuestra aplicación queda lista para ser ejecutada, al ejecutarla por primera ves nos mostrara la siguiente Figura

## CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE PROPIETARIO

The figure consists of two side-by-side screenshots of a Windows application window titled "Form1". Both screenshots show a table with columns "Id", "nombre", and "apellido".

**Screenshot (a): Agregando al usuario Jon Jeep**

Id	nombre	apellido
3	Steven	Gomez
4	Stephen	Duran
5	Hans	Milan
6	Alexis	Duarte
7	Alexander	Garibay
8	Jonatan	Arce
9	Daniel	Gamez
*	Jon	Jeep

Below the table are input fields: "Nombre: Jon" and "Apellido: Jeep". There is also a blue "Agregar" button. At the bottom are buttons for "ID" (with a text input field) and "Eliminar".

**Screenshot (b): Usuario Jon Jeep agregado**

Id	nombre	apellido
3	Steven	Gomez
4	Stephen	Duran
5	Hans	Milan
6	Alexis	Duarte
7	Alexander	Garibay
8	Jonatan	Arce
10	Jon	Jeep

Below the table are input fields: "Nombre: " and "Apellido: ". There is also a blue "Agregar" button. At the bottom are buttons for "ID" (with a text input field) and "Eliminar".

Figura 6.17: Agregando usuario

6.17a, en esta también se muestran los usuarios existentes en la base de datos, también se observa que hay un nuevo usuario que esta siendo dado de alta al dar click en el botón Agregar se mostrara el resultado de la Figura 6.17b, que ya contiene al nuevo usuario.

Finalmente solo resta probar el funcionamiento del botón eliminar para ello eliminaremos al usuario Daniel Gamez que tiene el ID=9, la Figura 6.18a muestra lo anterior. En la Figura 6.18b se muestra como el usuario a sido eliminado.

**CAPÍTULO 6. DESARROLLO DE APLICACIONES UTILIZANDO  
SOFTWARE PROPIETARIO**

---

The figure consists of two side-by-side screenshots of a Windows application window titled "Form1". Both screenshots show a table of users with columns "Id", "nombre", and "apellido".

**Screenshot (a): Eliminando al usuario Daniel Gamez**

Id	nombre	apellido
3	Steven	Gomez
4	Stephen	Duran
5	Hans	Milan
6	Alexis	Duarte
7	Alexander	Garbay
8	Jonatan	Arce
9	Daniel	Gomez

Below the table are input fields: "Nombre" (text box), "Apellido" (text box), and "Agregar" (button). At the bottom are "ID" (text box containing "9") and "Eliminar" (button).

**Screenshot (b): Usuario Daniel Gamez eliminado**

Id	nombre	apellido
2	Eduardo	Cota
3	Steven	Gomez
4	Stephen	Duran
5	Hans	Milan
6	Alexis	Duarte
7	Alexander	Garbay
8	Jonatan	Arce

Below the table are input fields: "Nombre" (text box), "Apellido" (text box), and "Agregar" (button). At the bottom are "ID" (text box) and "Eliminar" (button).

Figura 6.18: Eliminando usuario

## Capítulo 7

# Desarrollo de aplicaciones utilizando software libre

### 7.1. Creación de un nuevo proyecto

Para la creación de la Solución en MonoDevelop solo es necesario dar click en el apartado “Start New Solution” o a través de la barra de herramientas “File->New->Solution” en la Figura 7.1 se muestra la primera opción.

El segundo paso es seleccionar el tipo de aplicación que en este caso es una aplicación de escritorio para ello seleccionamos una solución en C# y elegimos un proyecto GTK#, como se muestra en la Figura 7.2.

A continuación hay que seleccionar la versión de GTK que nos sea mas útil, en este caso gtk 2.10 es la versión mas adecuada para este proyecto, como se muestra en la Figura 7.3.

Finalmente tenemos el proyecto creado y MonoDevelop nos muestra la Figura 7.4, que es la vista previa a la solución del proyecto GTK creado anteriormente y en esta solo se observa el diseño se una ventana en blanco.

### 7.2. Creación de la base de datos.

En la parte de la creación de la base de datos es necesario instalar algún servidor de base de datos, y este se administrara de manera externa a la solución del proyecto para ello fe necesario instalar MySQL [1] y en este caso también se utilizo un administrador de base de datos llamado MySQL Workbench [2].

Dentro MySQL workbench se creo una nueva base de datos llamada “IT-MDB” para ello seleccionamos “Create a new scheme in the connected server” como se muestra en la Figura 7.5, posterior a esto nos abrirá un nuevo menú en el que abra de dar el nombre de la base de datos que en nuestro caso es “IT-MDB” tal como se muestra en la Figura 7.6, al dar aplicar nos desplegará una nueva ventana en la que solo al que dar nuevamente aplicar como de muestra

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

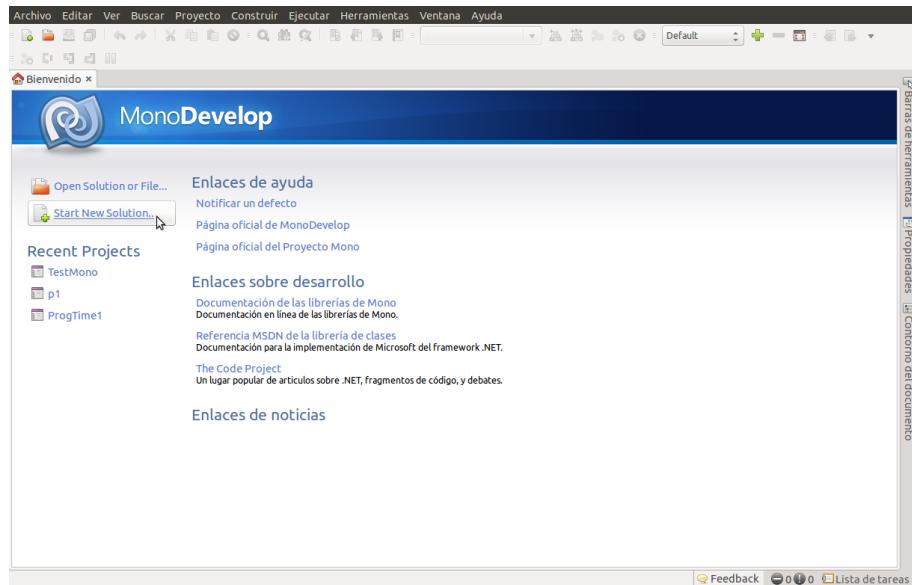


Figura 7.1: Crear nueva solución

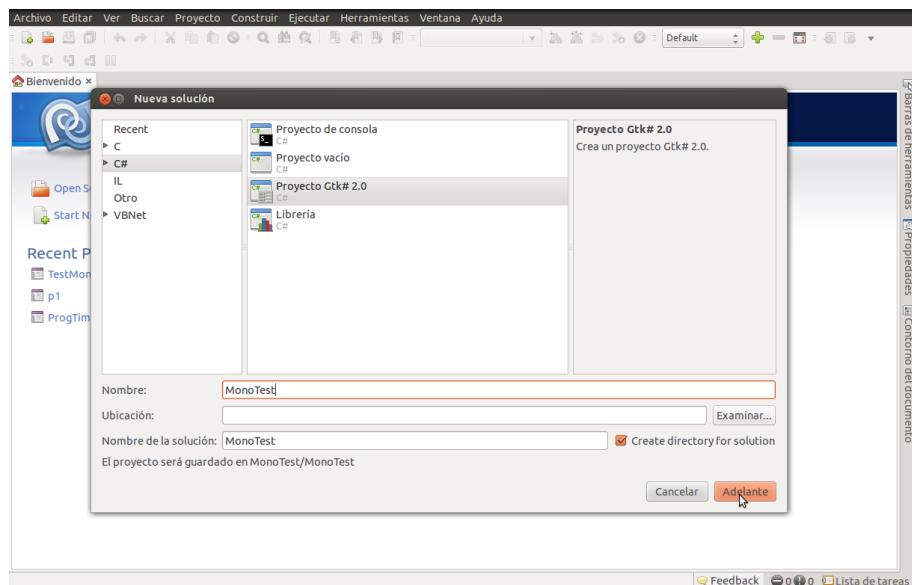


Figura 7.2: Selección de proyecto GTK

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

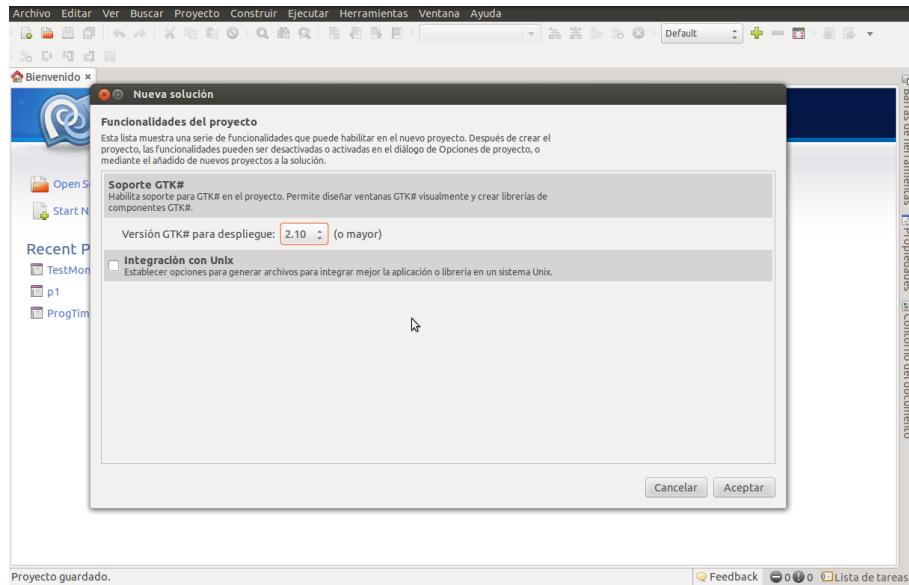


Figura 7.3: Selección de GTK apropiado

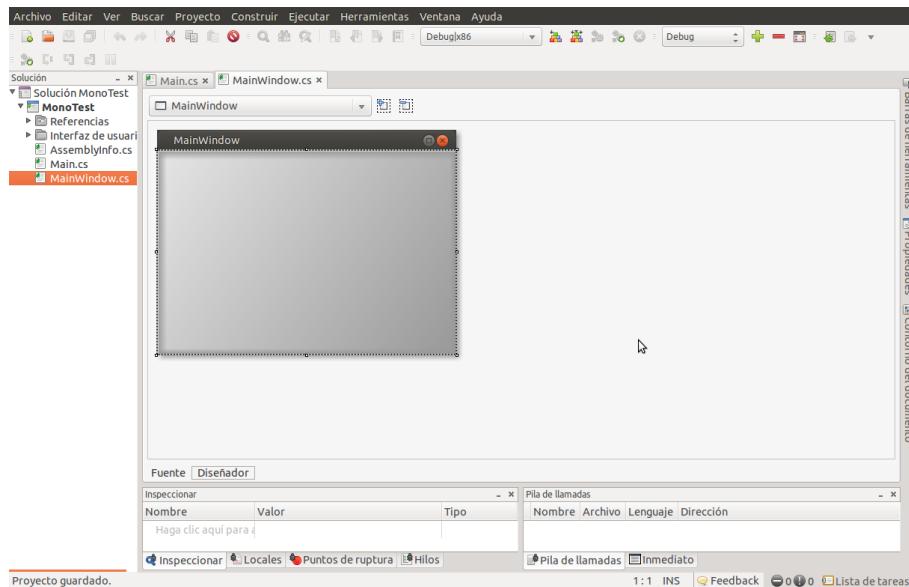


Figura 7.4: Diseño inician del proyecto GTK

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

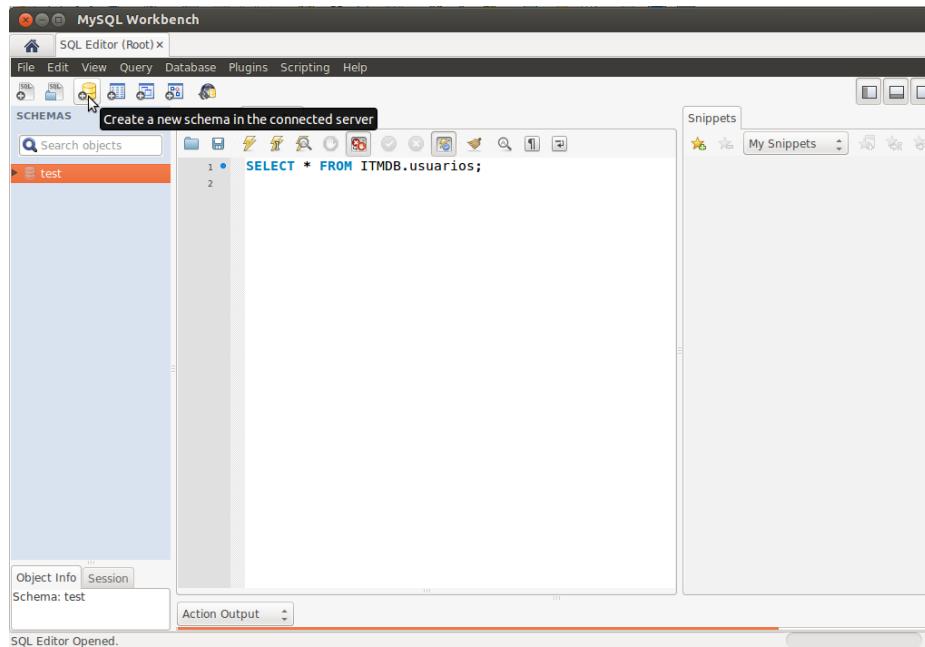


Figura 7.5: Creación de base de datos MySQL

en la Figura 7.7.

Posterior a esto se crea la tabla con el nombre “usuarios”, para esto daremos click sobre el apartado llamado “Create a new table in the active schema in connected server” como se muestra en la Figura 7.8, con los siguientes datos “ID, Nombre, Apellidos”, como se muestra en la Figura 7.9, al dar en aplicar nos desplegara una nueva ventana en la que muestra la sentencia que se ejecutara como se muestra en la Figura 7.10 a esta solo hay que dar aplicar para finalizar con la creación de la tabla “usuarios”.

A continuación, para poder realizar los experimentos fue necesario agregar datos a la tabla anteriormente creada para ello es necesario dar click derecho sobre la tabla usuarios este nos desplegará algunas opciones, en este menú daremos click en la opción “Edit Table Data”, al realizar esta opción podremos ir agregando datos a nuestra base de datos, en la Figura 7.11 muestra los campos de la tabla usuarios y a esta se agregan algunos datos, al finalizar solo daremos click en “Apply” para aplicar los cambios realizados a la tabla. con todo esto estará listo para continuar con la creación de la aplicación.

### 7.3. Creación de la ventana principal

En esta parte nos encargaremos del aspecto visual de nuestra aplicación. Para realizar esto tendremos que tener en modo diseño el archivo llamado “Main-

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

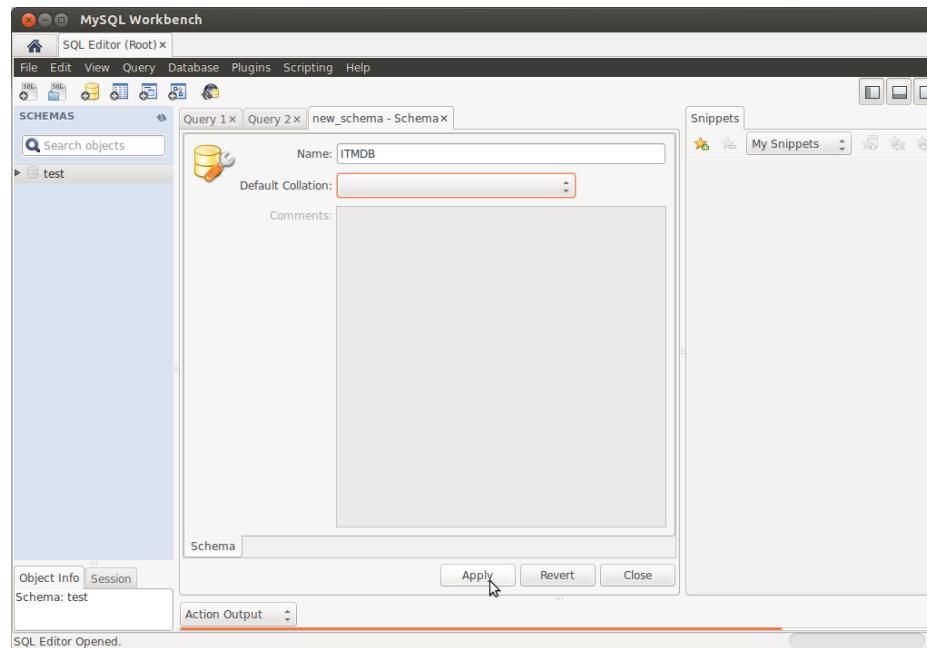


Figura 7.6: Nombrando la Base de Datos

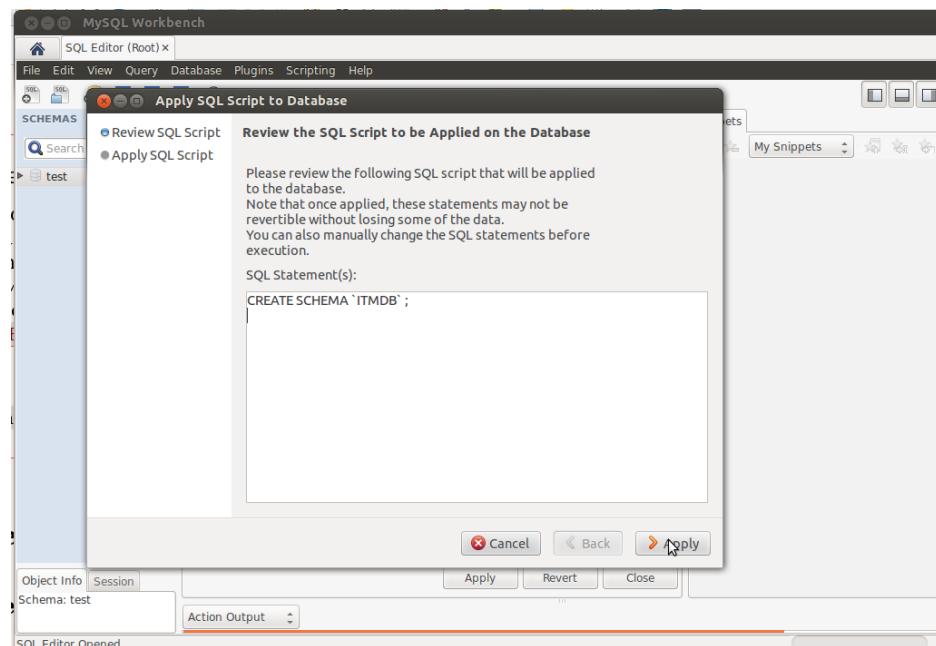


Figura 7.7: Finalizando la creación del esquema ITMDB

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

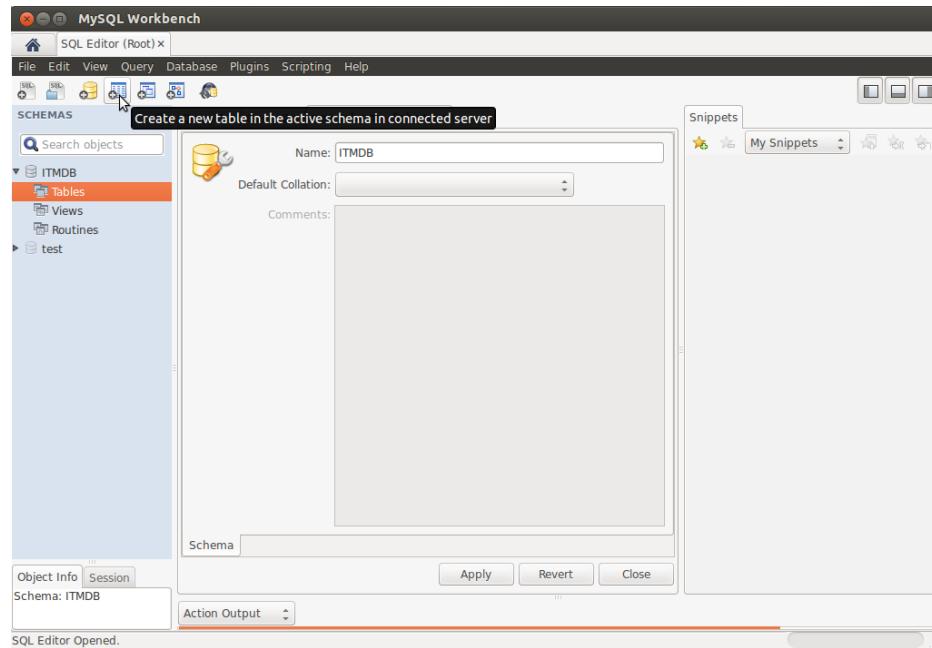


Figura 7.8: Creando Tabla usuarios en MySQL

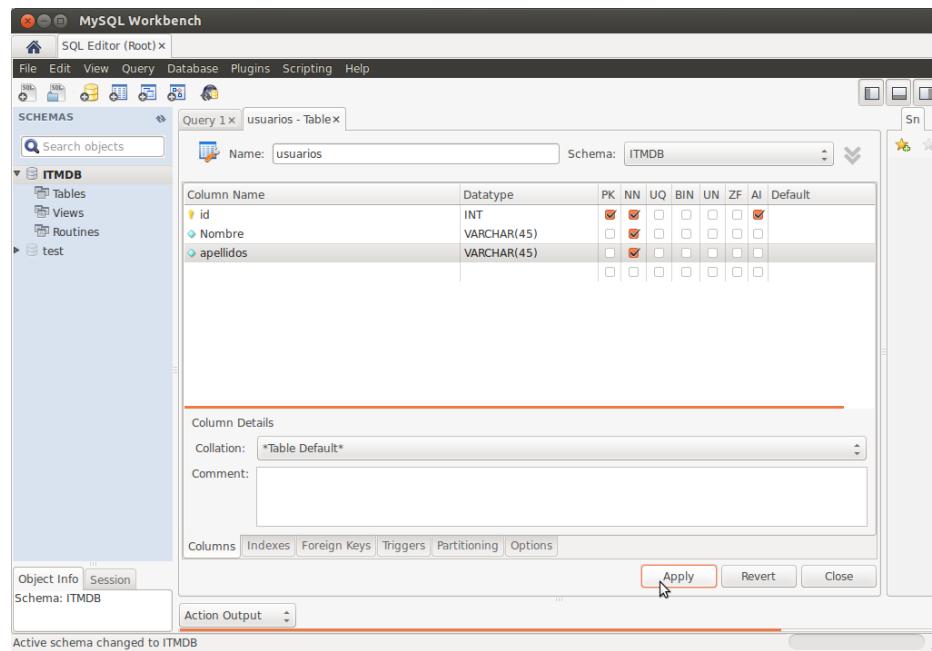


Figura 7.9: Nombrando Tabla en MySQL

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

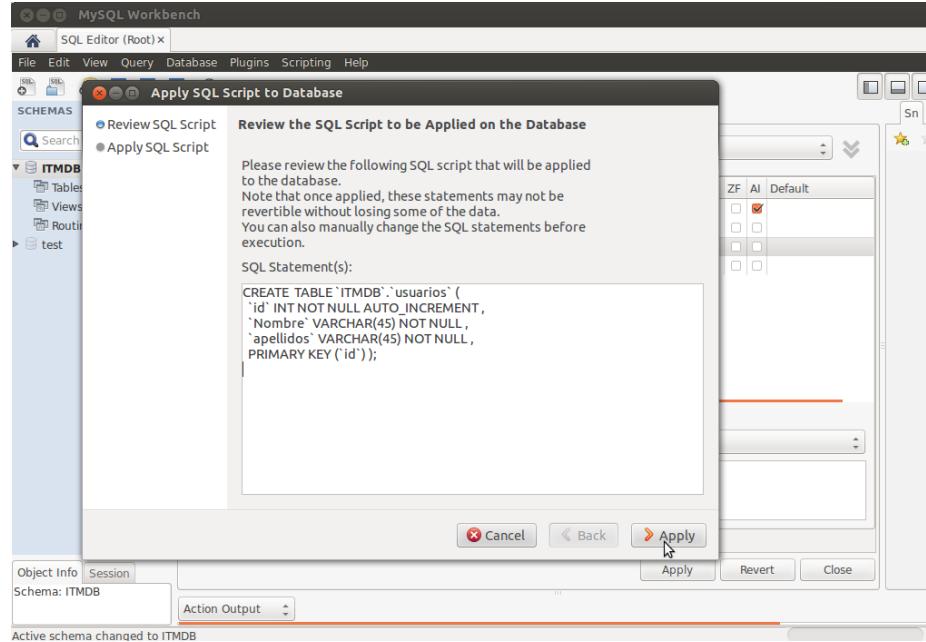


Figura 7.10: Finalizando la creacion de la Tabla usuarios

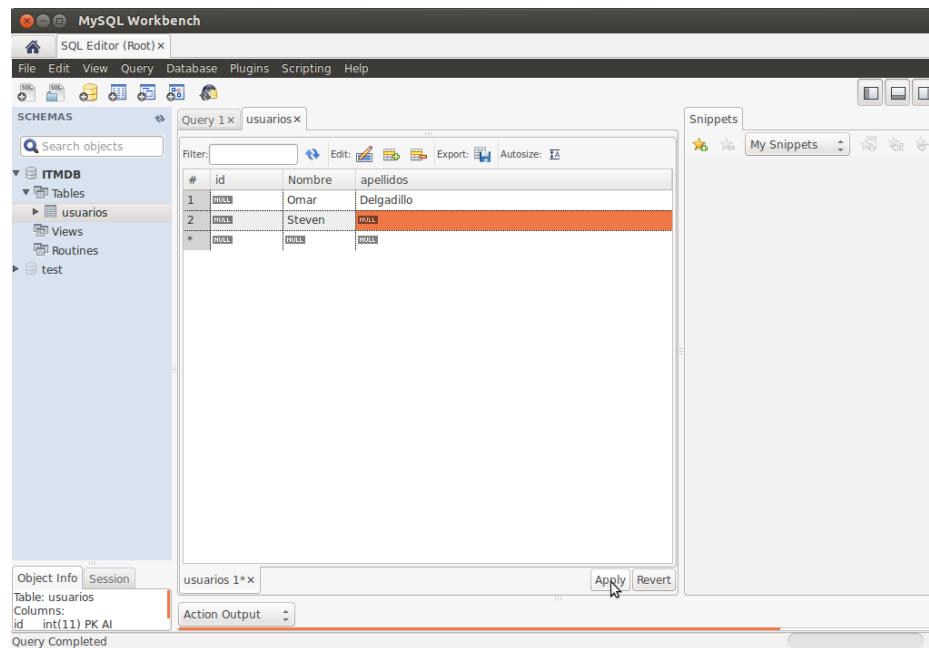


Figura 7.11: Insertando datos a la tabla usuarios

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

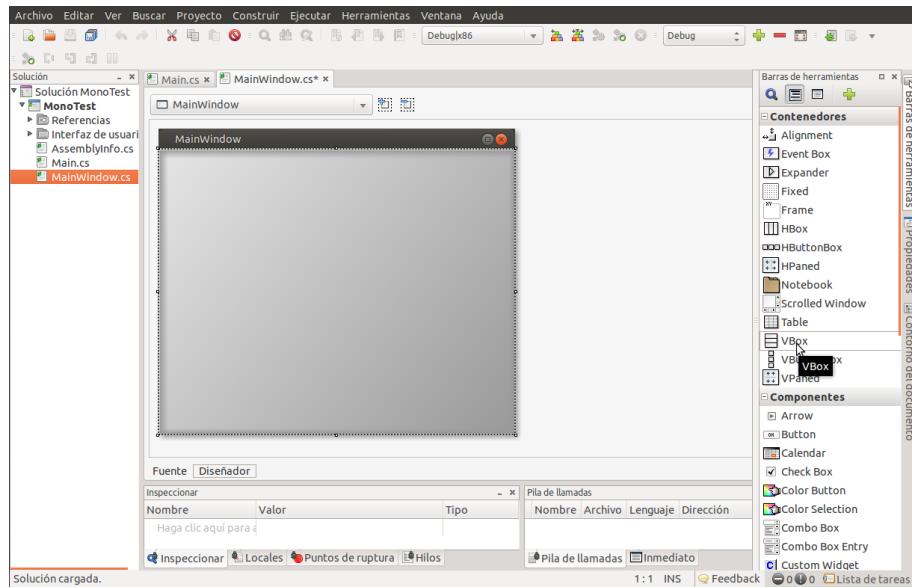


Figura 7.12: Agregando VBox a MainWindows.cs

Windows.cs”, seguido de esto iremos con nuestro ratón al lado izquierdo donde aparecen las opciones, barras de herramientas, etc.. en esta parte seleccionamos barras de herramientas y esta nos despliega todo con lo que podemos crear el aspecto de muestra aplicación en este como primera opción seleccionamos “VBox” como se muestra en la Figura 7.12, el cual nos divide de manera horizontal en 3 nuestra ventana. El siguiente paso es agregar un “Scrolled window” dentro de la primera sección en el “VBox” como se muestra en la Figura 7.13, después es necesario agregar un “Tree View” como se muestra en la Figura 7.14 en la misma parte en donde se coloco “Scrolled window”, en el “Tree View” es donde mostraremos la información de la base de datos.

En la segunda y tercera sección en el “VBox” es necesario agregar un “HBox”, y nuestra ventana principal quedara como se muestra en la Figura 7.15, de manera regular al agregar este elemento a la ventana solo se integra con 3 separaciones debido a esto nos vemos obligados a agregar 2 nuevos campos de la siguiente manera sobre la segunda sección del “VBox”, click derecho sobre la segunda sección nos desplegará un menú vamos a HboxX aquí daremos click sobre insert after o insert before, donde X sera correspondiente al numero de Hbox insertado sobre la segunda sección y nuestra ventana quedara como se muestra en la Figura 7.16.

En el primer contenedor de nuestro primer “HBox” que esta formado por 5 contenedores, colocaremos una etiqueta o “Label”, y esta quedara como se muestra en la Figura 7.17, seguidamente en el segundo contenedor colocaremos un “Entry” en la Figura 7.18 observamos como nuestra aplicación se encuentra tomando forma. En el tercero y cuarto contenedor agregaremos un “Label” y un

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

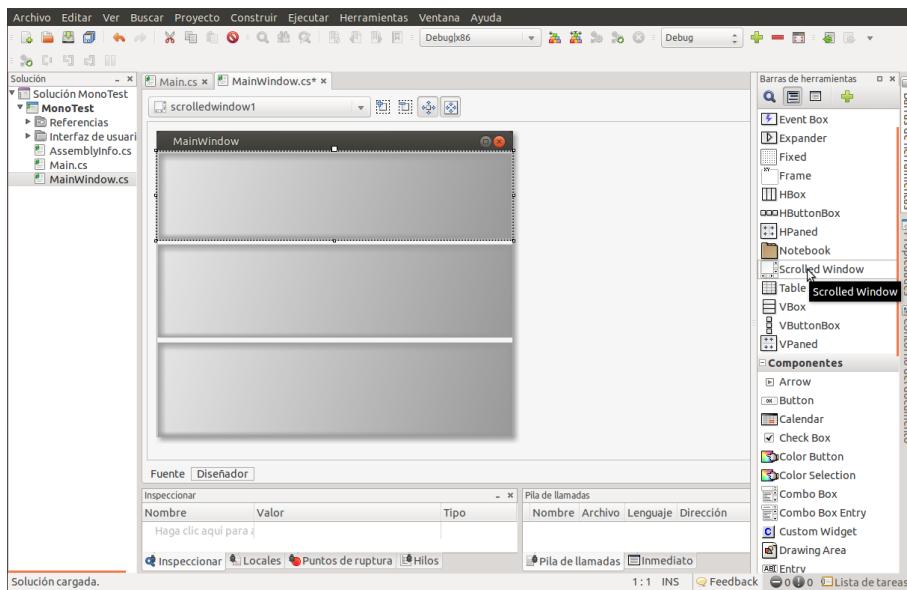


Figura 7.13: Agregando Scrolled windows a VBox

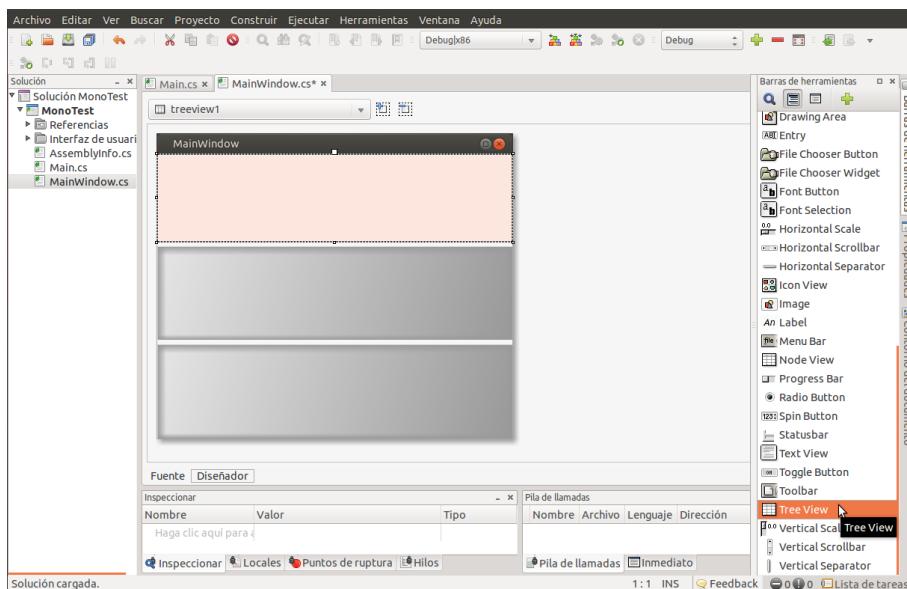


Figura 7.14: Agregando Tree View en Scrolled window

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

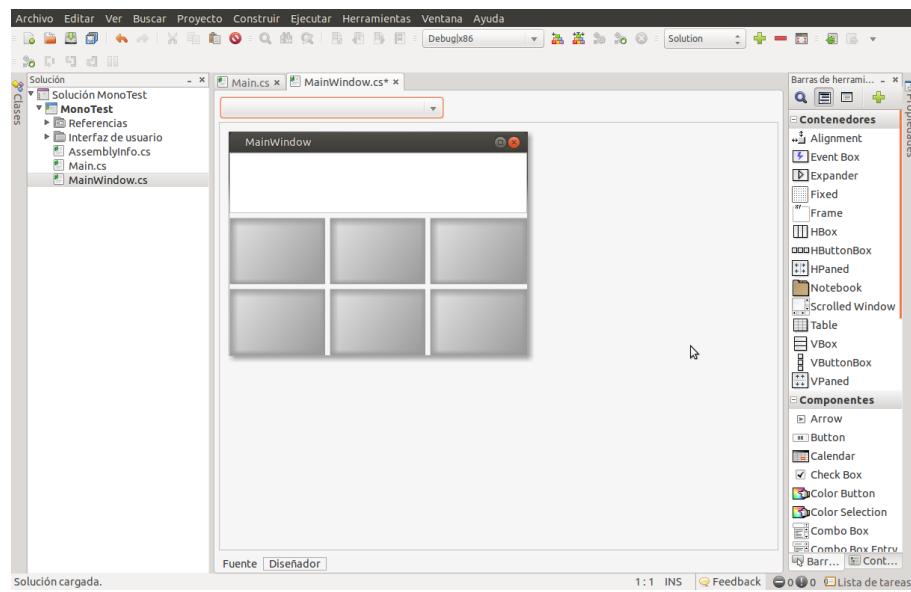


Figura 7.15: Agregando HBox en VBox

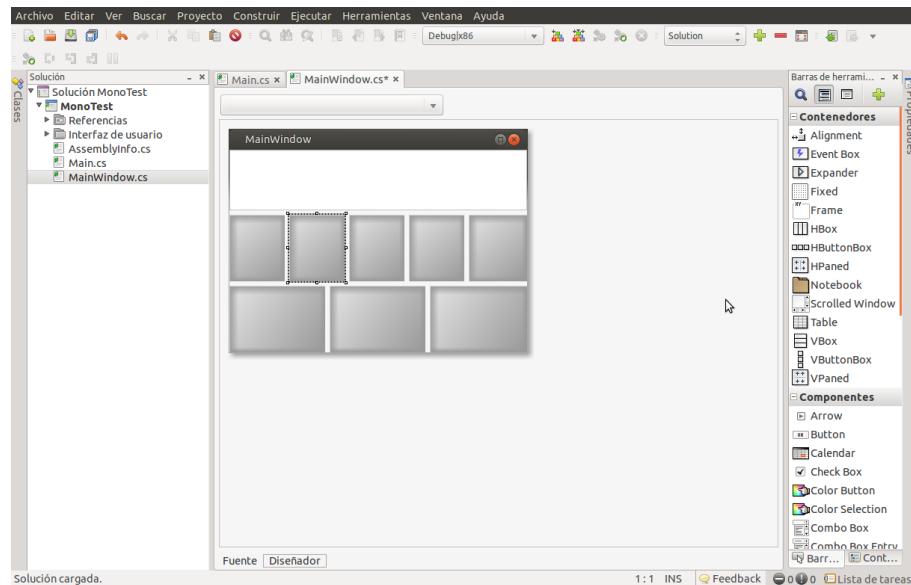


Figura 7.16: Agregando nuevo contenedor a HBox

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

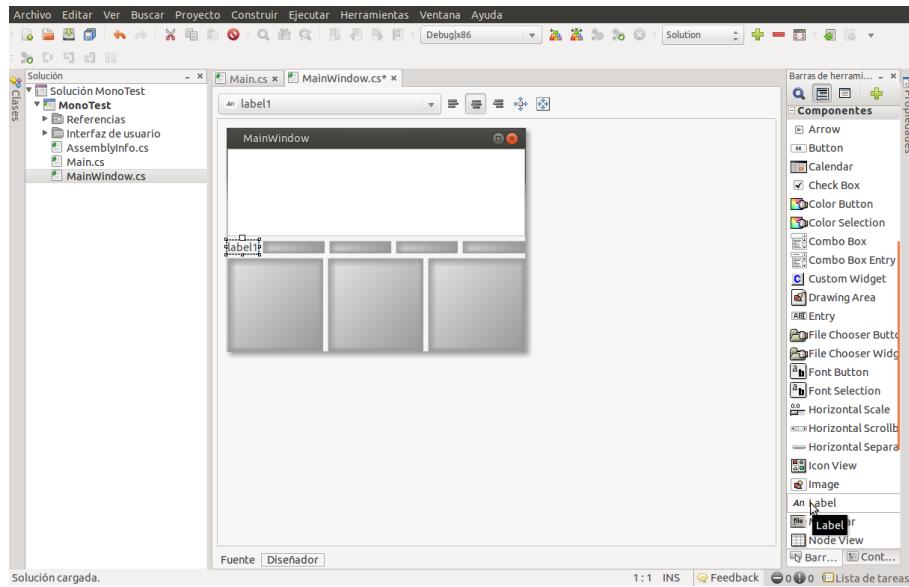


Figura 7.17: Agregando Label1

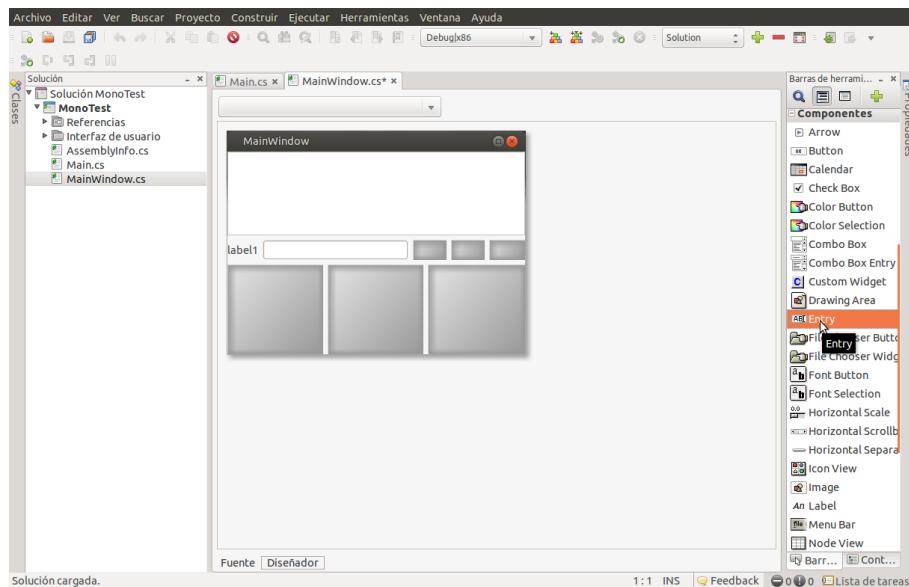


Figura 7.18: Agregando Entry1

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

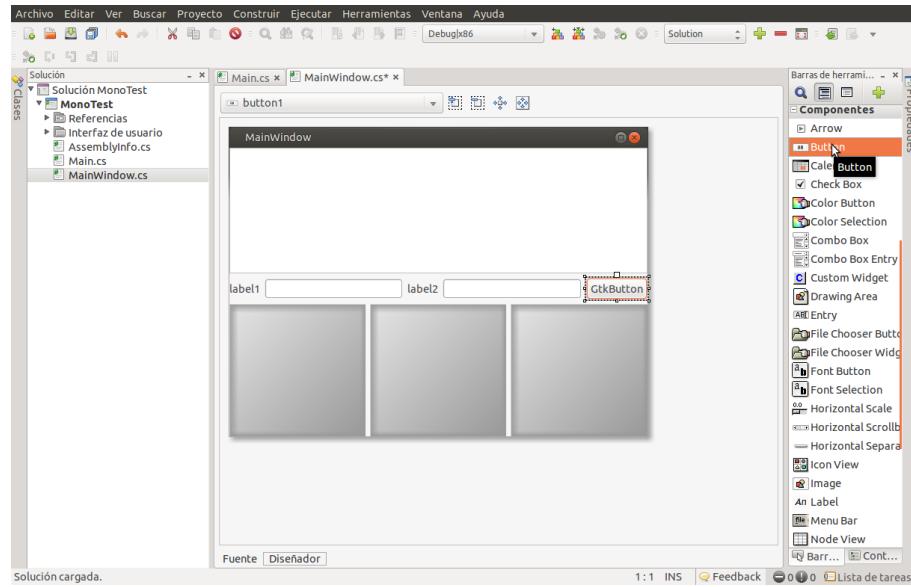


Figura 7.19: Agregando Button1

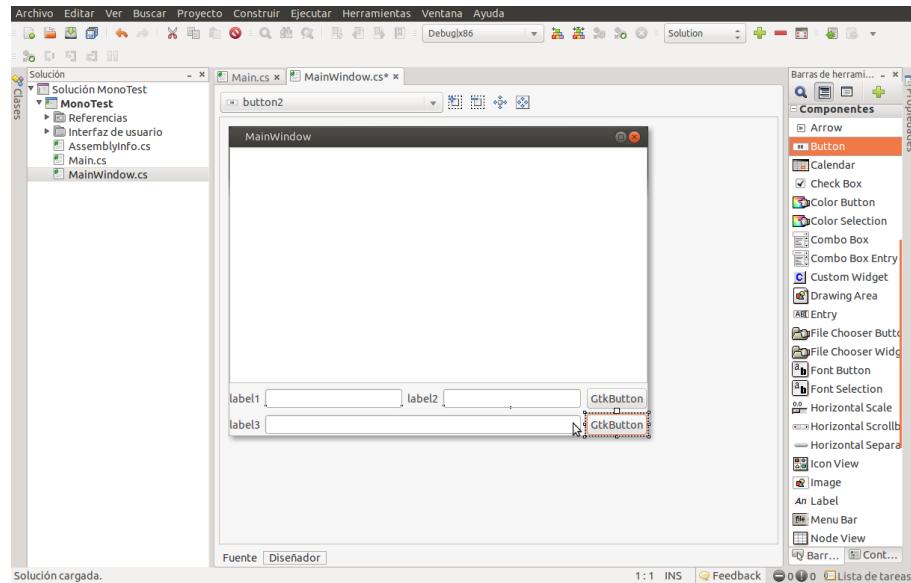


Figura 7.20: Agregando Button2

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

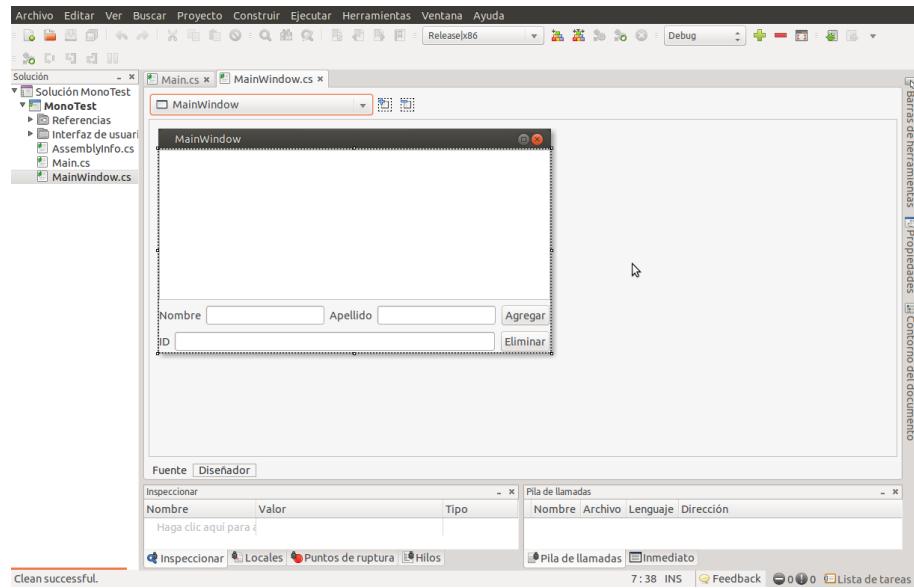


Figura 7.21: Ventana terminada

“Entry”, y en el quinto contenedor colocaremos un botón “Button” y nuestra ventana quedara como se muestra en la Figura 7.19. En el segundo HBox que esta formado de 3 contenedores agregaremos un “Label”, un “Entry” y un “Button”, la Figura 6.15 muestra la forma de la ventana en estos momentos. Finalmente cambiamos los nombres de las etiquetas y botones, nuestra ventana quedara como se muestra en la Figura 7.21.

### 7.4. Agregar funcionalidad a los botones

En este momento la aplicación solo mostrara los elementos de la base de datos pero no podremos realizar ningün alta o baja ya que los botones aun no tienen funcionalidad. Para agregar la funcionalidad a nuestro botonesaremos lo siguiente:

1. Para el primer botón “Agregar”, estado en modo diseño iremos al apartado propiedades que se encuentra al lado derecho de la pantalla, hay seleccionamos la pestaña señales, iremos a Button Signals → Clicked, en el apartado *Manejador* daremos el nombre *AgregarClick* como se muestra en la Figura 7.22, al dar enter nos aparecerá el siguiente código:

```
protected void AgregarClick(object sender, System.EventArgs e)
{
```

}

Finalmente para que nuestro botón funcione perfectamente hay que agregar la conexión a la base de datos, y hacer la consulta adecuada tomando los valores de las “Entry” correspondientes, a continuación se muestra el código correspondiente para la funcionalidad de nuestro botón “Aregar”:

```
protected void AgregarClick(object sender, System.EventArgs e)
{
    IDbCommand comando = conn.CreateCommand();
    conn.Open();
    String strSQL = "INSERT INTO usuarios(nombre, apellidos)
VALUES " + ("'" + entry1.Text + "','" + entry2.Text + "'");
    comando.CommandText = strSQL;
    comando.ExecuteNonQuery();
    conn.Close();
    treeview_load();
    entry1.Text = "";
    entry2.Text = "";
}
```

2. Para el botón “eliminar”, de la misma manera que creamos el evento para el botón “Aregar”, estado en modo diseño iremos al apartado propiedades que se encuentra al lado derecho de la pantalla, hay seleccionamos la pestaña señales, iremos a Button Signals → Clicked, en el apartado Manejador daremos el nombre “EliminarClick”, al dar enter nos aparecerá el siguiente código:

```
protected void EliminarClick(object sender, System.EventArgs e)
{
}
```

Finalmente se agrega al código la conexión, se ejecuta la sentencia Delete con el “Entry” indicado, enseguida se muestra el código correspondiente para la funcionalidad de nuestro botón “Eliminar”:

```
protected void AgregarClick(object sender, System.EventArgs e)
```

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

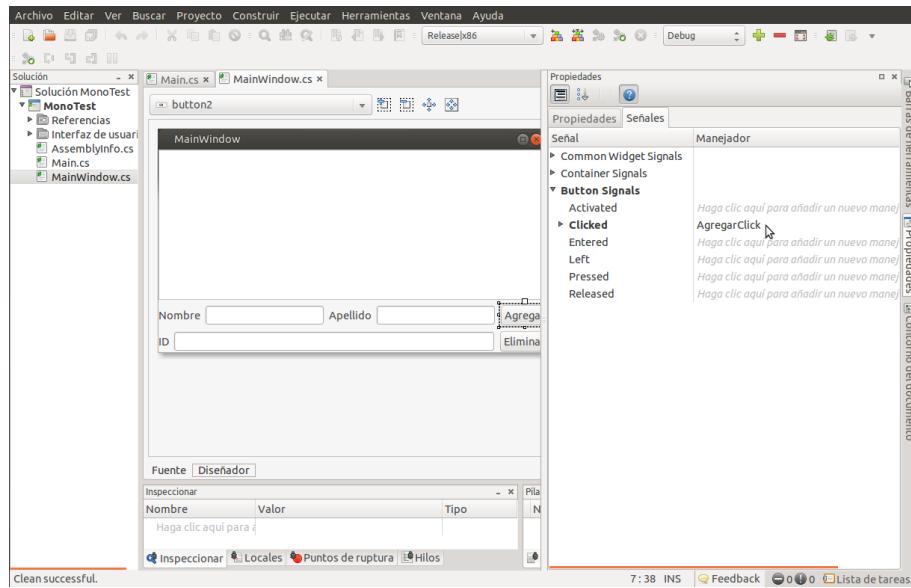


Figura 7.22: Agregando evento al botón Agregar

```
{  
    IDbCommand comando = conn.CreateCommand();  
    conn.Open();  
    String strSQL = "DELETE FROM usuarios WHERE id = "  
    +entry3.Text;  
    comando.CommandText = strSQL;  
    comando.ExecuteNonQuery();  
    conn.Close();  
    treeview_load();  
    entry3.Text = "";  
}
```

### 7.5. Aplicación en ejecución

Con todo lo anterior la aplicación queda lista para su ejecución, al ejecutarla por primera vez nos mostrara la siguiente Figura 7.23a, esta ventana nos muestra los usuarios dados de altas en la base de datos en ese momento, en esta misma se da de alta un nuevo usuario al dar click nos mostrara un contenido como el de la Figura 7.23b, en la Figura 7.24a muestra como se dieron de alta varios usuarios y podemos notar que tenemos un elemento a eliminar, al dar click al

## CAPÍTULO 7. DESARROLLO DE APLICACIONES UTILIZANDO SOFTWARE LIBRE

The figure consists of two side-by-side screenshots of a Windows application window titled "MainWindow".

**(a) Agregando usuario Steven**: Shows the initial state of the application. The table contains one row: "1 Omar Delgadillo". Below the table are input fields for "Nombre" (Steven), "Apellido" (Quezada), and "ID" (empty). Buttons for "Agregar" and "Eliminar" are visible.

**(b) Nuevo usuario agregado**: Shows the state after adding a new user. The table now contains two rows: "1 Omar Delgadillo" and "3 Steven Quezada". The input fields remain the same, but the "Agregar" button is highlighted in orange.

Figura 7.23: Agregando al usuario

The figure consists of two side-by-side screenshots of a Windows application window titled "MainWindow".

**(a) Eliminando al usuario Steven**: Shows the state before deletion. The table contains five users: "1 Omar Delgadillo", "3 Steven Quezada", "4 Eduardo Cota", and "5 Fabian Murrieta". The input fields show the values for Steven: "Nombre" (empty), "Apellido" (empty), and "ID" (3). The "Eliminar" button is highlighted in orange.

**(b) Usuario Steven eliminado**: Shows the state after deletion. The table now contains four users: "1 Omar Delgadillo", "4 Eduardo Cota", and "5 Fabian Murrieta". The input fields are empty, and the "Eliminar" button is visible.

Figura 7.24: Eliminando usuario

botón “Eliminar”, se eliminara el usuario y esto nos dará como consecuencia la Figura 7.24b, que muestra la actualización de los usuarios restantes en la base de datos.

# Capítulo 8

## Análisis comparativo

Comenzaremos hablando desde el momento de la instalación de ambas tecnologías. La instalación de visual studio 2012 es tardada ya que en esta se instalan bastantes paquetes que integran la misma, esto la convierte en una herramienta robusta, debido a esto se requiere un espacio considerable en disco. Por otra parte se requiere configuración de tipo de usuario y un numero considerable de click sobre ventanas que solo muestran información sobre esta herramienta, con todo esto se considera una dificultad media.

por otra parte la instalación de monodevelop es mas sencilla basta con acceder a una terminal y ejecutar el siguiente comando:

```
$ sudo apt – get install monodevelop
```

la principal diferencia es que solo se instala en entorno de desarrollo y el marco de referencia mono, si se requiere algún otro paquete este se deberá realizar por separado, un ejemplo de esto si queremos realizar una aplicación ASP.Net tendremos que instalar un servidor XSP. El paquete en cuestión se llamará, dependiendo del repositorio, mono-xsp o mono-xsp2 en caso de estar disponibles ambos, los podemos instalar ejecutando desde terminal el siguiente comando:

```
$ sudo apt – get install mono – xsp mono – xsp2
```

a diferencia de visual studio, monodevelop no requiere gran espacio en disco y esta no requiere una configuración por parte del usuario ya que esta cuanta con una ya predefinida, con todo esto consideraremos que tiene una dificultad baja.

El interfaz de usuario en ambas herramientas es bastante amigable fácil de usar y todos sus elementos son de una accesibilidad sencilla y no compleja debido a esto la dificultad de ambas se considera baja.

En ambas tecnologías la creación de una solución se realiza de una manera sencilla la ventaja de visual studio es que ofrece una amplia variedad de lenguajes de programación que a diferencia de monodevelop cuanta con pocos lenguajes

## CAPÍTULO 8. ANÁLISIS COMPARATIVO

---

	.Net	Mono
Instalación	media	baja
Interfaz de usuario	baja	baja
Creación de una nueva solución	baja	baja
Acceso a bases de datos	baja	media
Diseño de interfaz	baja	baja
Costo	alto	bajo

Tabla 8.1: Dificultad presentada en las tecnologías .Net y Mono

integrados, aunque esto no la ase menos potente solo reduce a los lenguajes mas comunes o mas conocidos por la mayoría de los programadores por lo tanto se considera que los dos tienes en este aspecto una dificultad baja.

El acceso a las bases de datos la podemos considerar como sencilla para ambas plataformas, con la particularidad que en visual studio tomo tiempo la creación de la base de datos y la configuración de todos los elementos que se requieren, te lleva a una serie de ventanas que creo son un excesivas o no requeridas, aunque todo esto al final te crea una ambiente sencillo para el manejo de altas bajas modificaciones y consultas de la base de datos. Por otra parte monodevelop en la versión utilizada no contaba con un administrador de bases de datos integrado debido a esto la creación de la base de datos se tiene que realizar con una herramienta fuera de este y también es necesario integrar un conector para poder acceder a la base de datos, esto se realiza de una manera fácil y rápida, al momento de acceder a la base de datos se realiza de una manera sencilla solo ejecutando la consulta requerida en SQL. Con lo anterior la dificultad de acceso a la base de datos en visual studio se considera baja y en monodevelop se considera una dificultad media ya que requiere mayor conocimientos en bases de datos.

En diseño de nuestro interfaces en ambas herramientas es bastante simple basta con arrastrar los componentes necesarios y dar forma a nuestra interfaz con esto podemos decir que la dificultad para la creación de una interfaz en visual studio y en monodevelop es baja. El aspecto principal a considerar el es costo de las diferentes versiones de Visual studio 2012, la versión mas económica se encuentra al rededor de los 9,000 pesos y su versión mas completa se encuentra al rededor de los 170,000 pesos y estamos obligados a el uso de su sistema operativo. Por otro lado mono es una tecnología de uso libre y se puede usar bajo cualquier arquitectura de procesador y en cualquier sistema operativo.

En el Tabla 8.1 podemos resumir todo lo anterior donde se muestra la dificultad de cada uno de los elementos mencionados.

## Capítulo 9

# Conclusiones y trabajo futuro

Se puede decir que Mono así como su IDE MonoDevelop son tecnologías maduras y amigables para la creación de aplicaciones bajo el marco de referencia .Net y si queremos realizar cualquier tipo de aplicación basta con revisar la documentación del marco de referencia .Net y/o la documentación de marco de referencia Mono para poder lograrlo. La principal ventaja de Mono es la que su MonoDevelop no es tan robusto como Visual Studio pero esto no lo ase menos, otro aspecto importante de resaltar es que esta herramienta es rápida al compilar nuestras aplicaciones.

El trabajo futuro que se tiene planeado es realizar un análisis comparativo de estas herramientas para la creación de aplicaciones de tiempo real, es decir aquellas aplicaciones que tengas restricciones de tiempo y conocer cual es la mejor opción.

# Bibliografía

- [1] <http://www.mysql.com/>.
- [2] <http://www.mysql.com/products/workbench/>.
- [3] [www.eclipse.org](http://www.eclipse.org).
- [4] [www.microsoft.com/visualstudio](http://www.microsoft.com/visualstudio).
- [5] [www.monodevelop.com](http://www.monodevelop.com).
- [6] [www.netbeans.org](http://www.netbeans.org).
- [7] [www.windev.com.mx](http://www.windev.com.mx).
- [8] José A. Blakeley. Data access for the masses through ole db. 1996.
- [9] Fabio Mancinelli Chaumartin, Etienne Coumont and Olivier Grisel. Bridging mono/.net and java in the scribo project: The way to ui.ma.net. *RMLL (Rencontres Mondiales du Logiciel Libre)*, 2009.
- [10] Manuel Rodenas Galian. Diseno e implementacion de servicios web multiplataforma utilizando el framework .net. Master's thesis, UNIVERSIDAD POLITECNICA DE CARTAGENA, 2012.
- [11] Maurice Eggen Gerald Pitts Gregory J Blajian, Roger Eggen. Mono versus .net: A comparative study of performance for distributed processing. 2006.
- [12] ECMA INTERNATIONAL. Ecma-335: Common language infrastructure (cli). 2010.
- [13] A. Krishna Mohan. Active server pages. Technical report, Sri Venkateswara University College of Engineering Tirupati, 2000.
- [14] Ing. Jorge Oma MSc. Luz Marina Santos Jaimes. J2ee and. net platforms in the development of web services. *ISSN*, 1:125–132, 2009.
- [15] Luis Dídimo Pucha Rumancela. Analisis comparativo entre arquitecturas asp.net tradicional y asp.net mvc en el diseno y desarrollo de sistemas informaticos. Master's thesis, UNIVERSIDAD NACIONAL DE CHIMBORAZO FACULTAD DE INGENIERIA, 2012.

---

## BIBLIOGRAFÍA

- [16] Prof. K. J. Satao Sampada K. Satav, Prof. S. K. Satpathy. A comparative study and critical analysis of various integrated development environments of c, c++, and java languages for optimum development. *UNIASCIT*, 1:9–15, 2011.
- [17] CM Sperberg-McQueen E Maler T Bray, J Paoli. Extensible markup language (xml). 2006.
- [18] W3C. Simple object access protocol (soap). 2000.
- [19] David N. Gray John Hotchkiss Seth LaForge Andrew Shalit Toby Weinberg. Modern languages and microsoft component object model. 1998.
- [20] Brian Singer Steve Tran Wellson Fung, Mike McLaughlin. Microsoft disk operating system. 2005.

# Apéndice:Código de la aplicación escrita en MonoDevelop V 2.8.6.3

```
using System;
using Gtk;
using System.Data;
using MySql.Data.MySqlClient;

public partial class MainWindow: Gtk.Window
{
    IDbConnection conn;
    public MainWindow (): base (Gtk.WindowType.Toplevel)
    {
        Build ();
        conn = new MySqlConnection("Server=localhost;" + "Database=ITMDB;" +
+ "User ID=root;" + "Password=123456;" + "Pooling=false" );
        treeview1.AppendColumn("Id", new Gtk.CellRendererText(), "text", 0);
        treeview1.AppendColumn("Nombre", new Gtk.CellRendererText(), "text",
1);
        treeview1.AppendColumn("Apellido", new Gtk.CellRendererText(), "text",
2);
        treeview_load();
    }
    protected void treeview_load()
    {
        conn.Open();
        Gtk.ListStore liststoreUsuarios = new Gtk.ListStore (typeof(int), typeof(string),
typeof(string));
        treeview1.Model = liststoreUsuarios; IDbCommand comando = conn.CreateCommand();
        string strSQL = "SELECT id, nombre, apellidos FROM usuarios";
        comando.CommandText = strSQL; IDataReader dr = comando.ExecuteReader();
        while (dr.Read())
    {
```

```
liststoreUsuarios.AppendValues(dr["id"], dr["nombre"].ToString(), dr["apellidos"].ToString());
}
dr.Close();
conn.Close();
}
protected void OnDeleteEvent (object sender, DeleteEventArgs a)
{
Application.Quit (); a.RetVal = true;
}
protected void AgregarClick (object sender, System.EventArgs e)
{
IDbCommand comando = conn.CreateCommand();
conn.Open(); string strSQL = "INSERT INTO usuarios(nombre,apellidos)
VALUES " + "(" + entry1.Text + "','" + entry2.Text + ")";
comando.CommandText = strSQL;
comando.ExecuteNonQuery();
conn.Close();
treeview_load();
entry1.Text="";
entry2.Text="";
}
protected void EliminarClick (object sender, System.EventArgs e)
{
IDbCommand comando = conn.CreateCommand();
conn.Open();
string strSQL = "DELETE FROM usuarios WHERE id = " + entry3.Text
;
comando.CommandText = strSQL;
comando.ExecuteNonQuery();
conn.Close();
treeview_load();
entry3.Text="";
}
}
```

# Apéndice:Código de la aplicación escrita en Visual Studio 2012

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace TestVS
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: esta línea de código carga datos en la tabla 'iTMDataset.Table'
            Puede moverla o quitarla según sea necesario.
            this.tableTableAdapter.Fill(this.iTMDataset.Table);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            int id=Convert.ToInt16(this.textBoxID.Text);
            this.tableTableAdapter1.Delete(id);
            this.tableTableAdapter.Fill(this.iTMDataset.Table);
            this.textBoxID.Text = "";
        }
        private void button1_Click(object sender, EventArgs e)
```

```
{  
    this.tableTableAdapter.Insert(this.textBoxNombre.Text,  
    this.textBoxApellido.Text);  
    this.tableTableAdapter.Fill(this.iTMDataSet.Table);  
    this.textBoxNombre.Text = "";  
    this.textBoxApellido.Text = "";  
}  
private void fillByToolStripButton_Click(object sender, EventArgs e)  
{  
    try  
    {  
        this.tableTableAdapter.FillBy(this.iTMDataSet.Table);  
    }  
    catch (System.Exception ex)  
    {  
        System.Windows.Forms.MessageBox.Show(ex.Message);  
    }  
}
```