UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA.
UNAN-LEÓN.

FACULTAD DE CIENCIAS.

DEPARTAMENTO DE COMPUTACIÓN.

CREACIÓN DE INTERFACES GRÁFICAS CON LA HERRAMIENTA DE DESARROLLO MONODEVELOP UTILIZANDO LENGUAJE DE PROGRAMACIÓN C# BAJO PLATAFORMA LINUX.

TRABAJO MONOGRÁFICO PARA OPTAR AL TÍTULO DE
INGENIERO EN SISTEMAS DE INFORMACIÓN.

Presentado por:

- Br. Marlin José Rivera Sánchez.
- Br. Meyling Yunieth Silva Rivera.
- Br. Francisco José Silva Sánchez.

Tutor:
Ing. Valeria Mercedes Medina Rodríguez.

León, Nicaragua, Septiembre de 2008.

"A la Libertad por la Universidad"

Agradecimiento

Al finalizar nuestro trabajo de Investigación, como forma de culminar nuestros estudios profesionales, damos nuestro agradecimiento a:

Dios, nuestro padre celestial por proveernos la vida y de todo lo necesario para cumplir esta meta.

Nuestros padres, por habernos apoyado moral, emocional y económicamente a lo largo de nuestro desarrollo académico.

Los profesores, por brindarnos los conocimientos básicos para emprender nuestro futuro profesional.

Nuestro tutor, quien siempre estuvo presente en la evolución de este trabajo monográfico.

Todos aquellos que colaboraron, de una u otra forma, para la realización de nuestro trabajo.

¡Gracias.!

Dedicatoria

El presente trabajo Monográfico lo dedico a:

Dios nuestro Señor, por haberme permitido concluir mis estudios e iluminarme en todo momento y por ser fuente de inspiración y sabiduría.

La Virgen María, por su intercepción y cubrirme con su inmenso amor.

Mis padres, Lic. Victoria Sánchez Sandino y Dr. Francisco Rivera Salinas por el apoyo incondicional y el sacrificio que me han brindado en todo momento para seguir a delante y lograr alcanzar mis metas y ser una persona de bien en servicio de la sociedad.

Mis hermanos, Dra. Benita Centeno Sánchez y Saúl Centeno Sánchez, por el apoyo que me han brindado en el transcurso de mi vida.

Mis Abuelitas, Juana Sandino Darce y Carmen Salinas Hernández, por el apoyo necesario en todos los momentos de mi vida.

Todas las personas que me colaboraron, de una u otra forma brindándome, su apoyo para la realización de este trabajo.

Br. Marlin José Rivera Sánchez.

Dedicatoria

El presente trabajo Monográfico lo dedico a:

Dios nuestro señor, por brindarme la vida, la sabiduría, la inteligencia, la paciencia, el discernimiento y todo lo necesario para lograr culminar satisfactoriamente mis metas.

Mis padres, Benjamín Félix Silva Lagos y Lilliam Clarisa Rivera de Silva por proveerme su apoyo incondicional para poder desarrollarme y crecer como persona y como profesional.

Mi esposo, Armando René Rodríguez López por ser mi apoyo y amigo incondicional en todo momento.

Mis hermanos, Everth, Harold y Javier Silva Rivera por ayudarme y cuidarme cuando lo he necesitado.

Br. Meyling Yunieth Silva Rivera.

Dedicatoria

El presente trabajo Monográfico lo dedico a:

Dios nuestro señor, por brindarme la vida, la sabiduría, la inteligencia, la paciencia, el discernimiento y todo lo necesario para lograr culminar satisfactoriamente mis metas.

Mis padres, Jairo José Silva Benavides *y Maria del Rosario Sánchez Meléndez de Silva* por proveerme su apoyo incondicional para poder desarrollarme y crecer como persona y como profesional.

Mis hermanos *Jairo, Gilda, Elías, Oscar y Miriam Silva Sánchez* por ayudarme y cuidarme cuando lo he necesitado.

Mis Amigos, que siempre me han apoyado en este largo proceso en especial al Dr. Luís Rojas Rosales.

Br. Francisco José Silva Sánchez.

Índice

Índice de Figuras

I. Introducción

Nuestro trabajo monográfico se basó en la implementación del uso del diseñador de interfaces gráficas Monodevelop, ya que es de fácil manejo, posee disponibilidad del código fuente y por estar escrito en lenguaje C# es altamente portable.

En función del entorno de desarrollo Monodevelop proponemos como ejemplo central una aplicación “Registro y Control de clientes del gimnasio Acción Club”, y como anexos las aplicaciones “Hola Mundo y Visor de imágenes” en nuestra aplicación principal que sirve de intérprete entre la base de datos y el usuario, hacemos uso del gestor de base de datos MySql. Con esto damos un aporte a los programadores y diseñadores de sistemas acerca de la creación de interfaces gráficas con acceso a bases de datos utilizando software libre.

La aplicación le facilitará al usuario la manipulación de los datos que mantiene un sistema seguro, consistente y evitando la redundancia de los datos.

II. Antecedentes

Con el modelo actual se complica demasiado el desarrollo de aplicaciones, el diseño y la creación de proyectos con software libre. Lo ideal es contar con un entorno que nos permita desarrollar prototipos, reutilizar código y sobre todo hacer aplicaciones sencillas sin un esfuerzo significativo.

Por otra parte, la complejidad de los sistemas operativos y el número de versiones que coexisten concurrentemente es mayor. Ante esta situación parece sensato realizar una revisión de las herramientas actuales y de la arquitectura de desarrollo. Es aquí donde entra el proyecto MONO que pretende convertirse en la opción libre a la arquitectura .NET de Microsoft.

El proyecto tiene como objetivo crear una implementación libre de algunas herramientas y parte de la arquitectura de .NET. La arquitectura .NET ha sido propuesta por Microsoft. Parte de esta tecnología se basa en un estándar propuesto a la ECMA y lo interesante es que tiene ideas muy buenas. Realmente, son tan buenas que el mundo del software libre debe poder disponer de ellas sin que tengamos que esperar a que Microsoft lo haga. Este es el objetivo del proyecto MONO.

Actualmente el desarrollo de aplicaciones gráficas en el Departamento de Computación de la Facultad de Ciencias de la UNAN-León, se ha basado en la realización de aplicaciones bajo entornos no open source.

Además, el desarrollo de aplicaciones gráficas con la herramienta de desarrollo Monodevelop bajo entorno Linux, no cuenta con ninguna documentación en el Departamento de Computación de esta Universidad.

III. Justificación

El motivo que nos impulsa a realizar este trabajo monográfico fue la necesidad de desarrollar aplicaciones donde se haga uso de software libre, que facilite la reducción de los costos generados al realizar aplicaciones con licencia comercial y conocer un entorno de desarrollo gráfico que nos permita realizar aplicaciones en corto tiempo utilizando la plataforma Linux, pero, que a su vez sea portable dicha aplicación, por esto hemos elegido Monodevelop como herramienta para la creación de nuestro software, ya que cuenta con una API (Application Program Interface) que incluye métodos para acceder a bases de datos diseñadas en MySql, además incluiremos ejemplos sencillos “hola mundo” y “Visor de imágenes” donde demostramos la versatilidad de desarrollo de estructuras de datos tradicionales.

Este trabajo será de mucha utilidad a futuros programadores de .Net y nosotros mismos, ya que Monodevelop posee un entorno gráfico muy fácil de utilizar y lenguaje de programación C# que es uno de los lenguajes más potentes y portables.

IV. Objetivos

Objetivos Generales:

- Aportar conocimientos generales acerca del creador de interfaces gráficas Monodevelop que utiliza lenguaje de programación C#.

- Desarrollar aplicaciones básicas utilizando MonoDevelop con lenguaje de programación C#.

Objetivos específicos:

- Desarrollar un sistema que facilite el ingreso y control de los afiliados del gimnasio "Acción Club" utilizando el creador de interfaces gráficas Monodevelop y el gestor de bases de datos MySql.

- Utilizar las cualidades que posee el creador de interfaces gráficas Monodevelop, así como sus facilidades de uso.

- Brindar a los programadores y diseñadores de sistemas un ejemplo claro de la utilización de Monodevelop y el gestor de bases de datos MySql.

- Aplicar conocimientos de programación, análisis y diseño de sistemas para desarrollar la aplicación de "Registro y Control de clientes del gimnasio Acción Club".

V. Marco Teórico

Mono es un proyecto de implementación del Framework .NET de Microsoft utilizando código libre (open source), gestionado por Ximian y basado en las especificaciones definidas en ECMA.

Actualmente, el proyecto puede considerarse bastante avanzado en muchos aspectos, pues dispone de una versión compatible con la versión 1.1 del framework y tiene muy avanzadas características de la versión 2. Además se están desarrollando partes que no son específicamente de la plataforma, como ADO.NET, WinForms y ASP.NET.

5.1 Contexto del proyecto Mono

Mono fue concebido por Miguel De Icaza (Co-fundador de la empresa Ximian, fundador y presidente de la GNOME Foundation). Siendo el proyecto en aquel entonces patrocinado por su compañía Ximian; actualmente Novel es quien patrocina el proyecto Mono, ya que éste adquirió a Ximian. La motivación de crear Mono se debe a la búsqueda de herramientas que ayudaran a la creación rápida de aplicaciones en el entorno Linux.

GNOME siempre había luchado por proporcionar facilidades al programador y una de las características más conocidas es que existen multitud de bindings que permita utilizar cualquier lenguaje de programación que desarrolle aplicaciones. Pero la elaboración de dichos binding era tremendamente 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 realizo una implementación de componentes, llamada Bonobo, utilizando CORBA. Pero tampoco ha tenido éxito, ya que era necesario que todo el software utilizase esa característica y eso no fue así. Por tanto, con .NET se abre 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 consigue lo que el proyecto siempre había buscado, independencia del lenguaje para programar en dicho escritorio.

Miguel de Icaza, luego de analizar el intérprete del byte code, advierte que no existen especificaciones. En febrero de 2001 comienza a indagar por dicha información en las listas de correo de .NET y al mismo tiempo comienza a trabajar en un compilador C# en cooperación con Rhys Weatherley y Jay Freeman, el mismo fue programado en C# como un ejercicio para demostrar su potencia.

En abril de 2001, la ECMA pública el formato de archivos faltantes y en GUADEC (6 al 8 de abril de 2001) Icaza demuestra las habilidades de su compilador. Luego de un minucioso análisis, donde claramente se concluye que es posible construir esa tecnología, Ximian reasigna recursos humanos de otros proyectos y crea el equipo Mono. Aspirando a tener una herramienta que fuese un sustituto completo de la tecnología.NET, formaron **“The Mono Open Source Project”**, el cual fue anunciado en julio de 2001, en la conferencia de O'Reilly.

Pasaron 3 años hasta que el 30 de junio de 2004 Mono 1.0 finalmente fue lanzado. Aunque Monodevelop en un principio solo podía ejecutarse en Mac y distintas distribuciones de Linux, no era posible ejecutarlo sobre Windows. Sin embargo, un IDE llamado SharpDevelop facilita la compilación de aplicaciones Mono sobre ambiente Windows pero carece de capacidad para depurar dichas aplicaciones (según las características de SharpDevelop Versión 1.1). Las versiones 2.0 en adelante incorporan un depurador integrado.

La plataforma .NET, busca unos objetivos muy similares a los buscados por GNOME, ofrecer una independencia de lenguaje a los programadores, así mismo

es también más avanzada, documentada, más amplia en su ámbito de actuación y tiene un diseño consistente. Cualquier API que se escriba utilizando un lenguaje que genere código para el CLR puede usarse desde cualquier otro lenguaje que genere código para esta plataforma.

La plataforma creada por Microsoft tiene una serie de puntos muy interesantes para el proyecto, como son:

- Un entorno virtual de ejecución (CLR o VES), que provee recolección de basura, threading…
- Una librería de clases potente y comprensible
- Un nuevo lenguaje muy similar a C++ o Java (C#).
- Una especificación de lenguaje que pueden seguir los compiladores, con el fin de generar código que pueda interactuar con otros lenguajes de programación (CLS)

5.2 Comparativa de Mono con .NET

La iniciativa .NET de Microsoft es un proyecto de toda la compañía bastante difuso, una parte del cual es el framework de .NET. Mono es una implementación de este framework, pero de nada más relacionado con .NET, como Passport o software como servicios.

Mono no es un instrumento de desarrollo, como el de Microsoft Visual Studio. Más bien, es un puerto de la esencia que guardan a Microsoft en el desarrollo de herramientas. Esto incluye Microsofts C# que es un lenguaje de desarrollo, "bibliotecas" de código escrito previamente y de Microsoft el CLR (Common Language Runtime), software que permite a un programador para combinar el código escrito en diferentes idiomas en una sola aplicación.

Los objetivos iniciales del proyecto Mono eran implementar en un entorno de software libre para el mundo Unix la especificaciones ECMA, para lo cual se incluye un compilador de C#, un entorno de ejecución CLR y un conjunto de librerías de clase que incluyen las FCL, así como otras añadidas.

A continuación se muestra un gráfico que representa algunos de los diferentes tipos de tecnologías que se están implementando en Mono. Los elementos señalados en azul, son los correspondientes a ECMA y que no están bajo ningún tipo de patente.

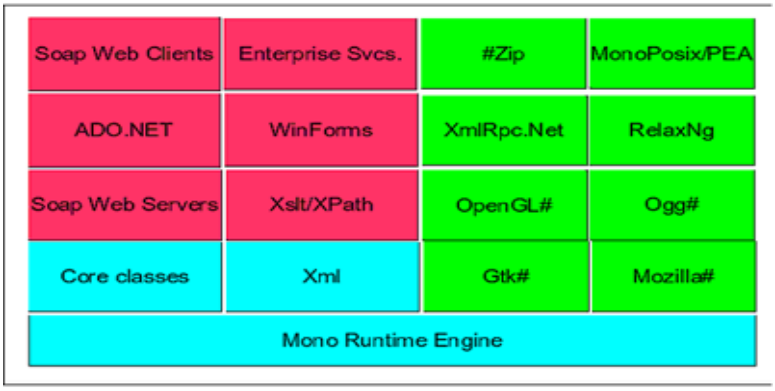

Fig. 1 Tecnologías de Mono

5.2.1 Características de mono

A continuación están algunas de las características más importantes de de Mono:

- Independencia de lenguaje: puedes usar clases escritas en cualquier lenguaje soportado por Mono (por ahora, C#, Mono Basic, Java, Nemerle, MonoLOGO, Boo, IronPython, GTK#).

- Independencia de plataforma: las aplicaciones son muy portables, y la mayoría compatibles en binario entre plataformas.

- Gran soporte para bases de datos: MS SQL, MySQL, Postgres, OLE DB, en total hasta 27 bases de datos.

- Velocidad: el lenguaje intermedio se compila en cada plataforma con unos compiladores muy rápidos (JIT) lo que lo hace mucho más rápido que lenguajes interpretados como PHP ó Python y más rápido en la compilación (JIT) que Java. Únicamente un poco más lento que C.

- Gestión automática de memoria: es una fuente inagotable de errores y se pierde una gran cantidad de tiempo programando esto. Si se automatiza, se obtiene más tiempo que se puede dedicar a resolver el verdadero problema.

- Seguridad.

- Aplicaciones Web: Cualquier lenguaje soportado por Mono se puede usar para Aplicaciones Web. No hay necesidad de lenguajes especiales como PHP.

- Servicios Web: soporte para SOAP.

- Soporte para XML: Mono tiene muchas clases para trabajar con xml.

- Extensa librería de clases: Criptografía, HTTP, Bases de datos, GUI, etc.

- Aplicaciones GUI multiplataformas: se pueden escribir aplicaciones con interfaz gráfica que se ejecutan invariablemente en multitud de plataformas. Por ejemplo, Gtk# es muy potente y disponible prácticamente en cualquier plataforma.

5.2.2 Mono Runtime

Sería el equivalente al Common Language Runtime o entorno virtual de ejecución. Implementa un compilador JIT para el CIL de la máquina virtual, un compilador Ahead-Of.-Time (AOT), un cargador de clases, un recolector de basura, el sistema de thereadings y las librerías de acceso a los metadatos

En estos momentos existen dos entornos de ejecución:

- Mono: compilador JIT y AOT. Únicamente soporta máquinas x86 por el momento.
- Mint: intérprete de mono. Es un entorno fácil de migrar a nuevas plataformas.

El entorno de ejecución se puede utilizar también embebido dentro de otra aplicación, de forma que se pueden extender aplicaciones C y C++ mediante C#. En comparación con la solución ofrecida por Microsoft, Mono incluye un intérprete y la opción de utilizar el runtime de forma embebida en una aplicación.

Mono funciona tanto sobre plataformas de 32 bits como de 64 bits: s390, SPARC, SPARCv9 (64 bits), PowerPC, x86, x86-64 (64 bits), así como sobre diversos sistemas operativos: Linux, MacOS X, BSD, SUN SOLARIS y Microsoft Windows. Aparte el intérprete puede ejecutarse sobre otras plataformas como HP-UX. Hay más información al respecto en la página del Mono runtime en el Web del proyecto.

El runtime es completamente operativo, aunque es posible que siga evolucionando en sentidos como optimizaciones para el compilador Just In Time o el recolector de basura. Dentro del runtime se soportan también mecanismos que permiten a las aplicaciones hechas en mono comunicarse con otras aplicaciones, o utilizar librerías externas.

- PInvoke: es el mecanismo que se utiliza desde el código gestionado para acceder a librerías externas. Se está utilizando en mono para acceder a llamadas de la API de Unix, así como para comunicarse con librerías del sistema.

- Remoting: Mono tiene soporte para Remoting y objetos Proxy. El entorno de ejecución facilita este tipo de servicios.

- COM and XPCOM: no se prevé añadir soporte para XPCOM en Unix y COM en Windows.

5.2.3 Class Library

Se ha buscado una compatibilidad total con la implementación .Net de Microsoft. La librería de clases se está desarrollando en C# y puede ser utilizada por cualquier lenguaje, gracias al Common Language Specification. Hay ya desarrollada una gran parte de las clases pertenecientes al Framework Class Library, lo que sumado a la madurez del runtime, hace que sea posible realizar gran cantidad de aplicaciones, por lo que es posible que encontremos algún bug. El proyecto dispone de una página en la que se informa del estado exacto de desarrollo de cada parte de la plataforma.

5.2.4 Compilador C#

El compilador de C# (MCS) de mono en estos momentos se considera que cumple la especificación, por lo que es un producto relativamente maduro. MCS está escrito en C# y es capaz de compilarse a si mismo desde enero de 2002, así como a otros muchos programas.

5.2.5 Otras tecnologías

En los siguientes puntos se comentan algunas tecnologías que están siendo desarrolladas, pero que no forman parte de la especificación ECMA.

- *Visual Basic*: Mono incluye un compilador para MonoBasic (Abas). Gracias a la independencia de lenguaje, el código escrito en MonoBasic puede ser utilizado desde C# y viceversa. Está actualmente en fase de desarrollo y queda bastante trabajo pendiente.

- *ADO.NET*: se dispone de providers para un buen número de bases de datos, superior de hecho al ofrecido por Microsoft. Todavía queda bastante trabajo por realizar, sobretodo en lo referente a completar los providers, realizar herramientas necesarias como xsd.exe, y terminar la librería de clases.

- *ASP.NET*: es una tecnología muy potente para la creación de sitios Web y servicios Web. Al igual que el resto, es independiente del lenguaje, por lo que podemos tener páginas escritas en Pascal, Logo, Basic, C#, C++, dentro de la misma aplicación. Las páginas son compiladas, por lo que su ejecución es muy rápida. Ofrece también un modelo de acceso a los elementos de la página mediante objetos, un manejo de los mismos a través de eventos, o un control del estado de la página, entre otras características interesantes. La implementación se divide en dos partes, Web Forms y Web Services. Ambas son funcionales en este momento. Pese a ser funcional, queda bastante trabajo para hacer la implementación compatible con la solución de Microsoft. ASP.NET puede funcionar con el servidor xsp.exe o con el módulo mod_mono de apache.

- *XSP*: es un pequeño servidor Web que se utiliza para ejecutar aplicaciones ASP.NET.

- mod_mono: es el módulo de conexión de apache con ASP.NET, de forma que se pueda utilizar apache para servir páginas ASP.NET. El proyecto está en curso y aunque funciona queda bastante trabajo pendiente.

- GTK#: es un conjunto de clases realizadas en C# que permiten el acceso a GTK+ y otras librerías que forman parte de GNOME. Funciona tanto sobre Linux como sobre Windows. Existen ya aplicaciones desarrolladas sobre esta tecnología, aunque aún queda trabajo pendiente.

5.2.6 Licencias

Mono usa tres tipos de licencia:

- Las librerías de clases están bajo una licencia X11. Este tipo de licencia permite prácticamente cualquier uso del código, incluido copiarlo y usarlo en una aplicación propia. El único requisito es que se mantenga la información de copyright de los archivos. No es necesario siquiera indicar que se está utilizando software bajo licencia X11.

- Las librerías del runtime, como el JIT, se distribuyen bajo licencia LGPL. De esta forma si se realiza un programa que linke con ellas, se puede mantener bajo código propietario. Sin embargo es necesario permitir que se puedan enlazar con versiones más recientes de las librerías. La forma más fácil de hacer esto sería linkando dinámicamente con ellas. Esto obligaría al usuario a descargar e instalar mono por separado.

- El resto de aplicaciones, como mono o mcs, usan una licencia GPL. Por lo tanto cualquier aplicación que estén basadas en ella, deberán mantener esta licencia.

En lo que respecta al cumplimiento de las patentes de software, las partes contempladas en el estándar ECMA no tienen ningún problema ya que se permite a cualquiera implementar esos componentes gratuitamente y para cualquier propósito. Sin embargo, cuestiones referentes a ADO.NET, ASP.NET y WinForms, son bastante diferentes. La estrategia de Mono ante estas tecnologías es la siguiente:

1. Evitar la patente utilizando otros mecanismos que permitan implementar la misma funcionalidad
2. Eliminar las porciones de código bajo patente
3. Encontrar algo más novedoso que deje sin uso a la patente.

5.2.7 Herramientas de desarrollo

Es posible desde Windows desarrollar una aplicación utilizando el Visual Studio. En el lado de Linux existe el Monodevelop, el cual es un IDE basado en el SharpDevelop que nos permite:

- Conectarse a bases de datos desde el IDE
- La inclusión de un debugger que permita al igual que el Visual Studio, ejecutar código línea por línea, y revisar valores de variables.

5.2.8 SharpDevelop

SharpDevelop es un entorno de desarrollo integrado (IDE) para la plataforma .NET. Soporta las versiones de Microsoft y de Ximian (MONO). Soporta desarrollo de interfaces, clases, namespaces y proyectos en C#, C++ .NET y VB.NET, además de permitir importar los proyectos creados con Microsoft Visual Studio .NET.

SharpDevelop, también conocido como #develop, es un editor de programación para proyectos con el que podemos desarrollar proyectos fácilmente, gracias a su

soporte para plantillas de diseño y para diversos lenguajes de programación, pues Incluye:

- Completado de Código
- Diseñador de Formularios
- Auto insertado de Código
- Conversor de Código (C#<-->VB.Net)
- Importar/Exportar Soluciones VS.NET a Visual Studio .NET
- Plegado de Código (*"Folding"*)
- Visor gráfico para realizar pruebas con NUnit
- Analizador del Código Ensamblador
- Vista previa de Documentación en XML
- Y mucho más: sintaxis coloreada, paréntesis inteligentes, bookmarks, plantillas, herramientas para expresiones regulares, asistentes, exportación HTML, visor de clases, integración con NDoc, integración con Nprof.

5.3 MonoDevelop

MonoDevelop es un entorno de desarrollo gráfico (IDE - Integrated Development Enviroment) para Mono. Es un proyecto que porta la herramienta SharpDevelop a Gtk#.

Monodevelop persigue los siguientes objetivos principales:

- Crear uno de los mejores entornos de desarrollo para sistemas Unix para C# y Mono.
- Dado que está escrito utilizando las librerías de Gtk#, lo más probable es que se añada funcionalidad para mejorar la experiencia de programación en Gtk#.
- Separarse lo menos posible de SharpDevelop: idealmente lo que se quiere es la mezcla del código de ambas aplicaciones en una sola (a través de

compilación condicional e interfaces) maximizar las contribuciones y la velocidad de desarrollo.

5.3.1 Características de MonoDevelop

Las principales características de Monodevelop son:

- Trabajo Personalizable, incluidas las teclas, diseños personalizados, y herramientas externas.
- Soporte para varios lenguajes, como C#, VB.NET y en C/C++, Boo y Java (IKVM) con soporte disponible por separado add-ins.
- Soporte a la conclusión de código y tipo de información sobre herramientas.
- Refactoring para simplificar las operaciones de cambios, como el cambio de nombre de los tipos y el tipo de miembros, que encapsula los campos, superando a los métodos o interfaces de la aplicación.
- Código de navegación, operaciones como saltar a la definición de variables y la búsqueda de clases derivadas.
- Interfaz Gráfica Fácil de usar para el diseño para aplicaciones GTK#, también soporte a la creación y gestión de bibliotecas (librerías) GTK# widget
- Integrado de control de versiones de código fuente, con soporte para Subversión.
- Integrado de unidad sobre la base de pruebas Nunit.
- Soporte a proyectos de ASP.NET, que permite proyectos Web pueden ser construidos y probados en XSP.
- La base de datos integrada de explorador y editor (beta).
- Integración con Monodoc, a la documentación de las clases.
- Soporte para los makefiles, tanto de generación y sincronización.
- Soporte para Microsoft Visual Studio proyecto formatos.

- Embalaje sistema que permite la generación de archivos tar, código fuente y los paquetes binarios.
- De la línea de herramientas para la construcción y gestión de proyectos.
- Soporte para los proyectos de localización.
- Extensible para añadir en la arquitectura.

5.3.2 Aplicaciones hechas para Mono.

- *Monodevelop*: Es una IDE para programar en Linux programas para Mono. La IDE esta hecha en C#.
- *F-Spot*: Programa para catalogar fotografías, además de poder hacer algunas modificaciones digitales a las fotos.
- *Beagle*: Herramienta que indexa y busca información entre una serie diferentes tipos de documentos en Linux.
- *Tomboy*: Programa para almacenar notas que se ligan mediante palabras claves.
- *Muine*: Es un reproductor de audio basado en GStreamer.
- *PyMusique*: Programa que provee de una interfase gráfica que accede a comprar música con el servicio iTunes de Apple.
- *MonoUML*: Es un editor para realizar diagramas con el estándar UML.

MonoDevelop viene con una buena cantidad de widgets que podemos usar para el diseño de formularios llamada **Widget Palette** (paleta de componentes).

Widget: Un widget es todo lo que aparece en la pantalla, un botón, una ventana, una barra de desplazamiento, un menú, etc.

El widget es el átomo de la interfaz del usuario: recibe eventos de ratón, teclado y otros del sistema, y pinturas, una representación de sí mismo en la pantalla.

Cada widget es por lo general rectangular, y ellos se ordenan en un Z-orden. Un widget se sujeta por su padre y por el widgets delante de él.

Widgets de herramientas: Son los elementos que se utilizan para el desarrollo de formularios.

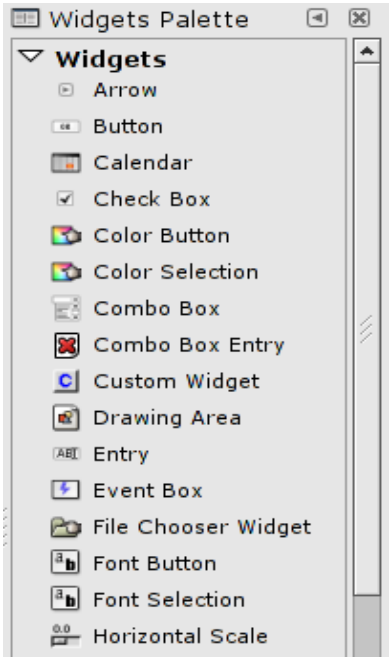

Fig. 2 Widgets de propiedades

Widgets de Propiedades: Se utilizan para establecer las propiedades de los componentes del formulario (Ver figura 77 de Anexos).

Widgets de Señales: Se utilizan para establecer los eventos que generaran los componentes del formulario (Ver figura 78 de Anexos).

Widgets básicos: se diseñan para el uso directo.

Button: El botón de orden o comando, es probablemente el widget más usado en

cualquier interfaz gráfica de usuario. El botón se utiliza para ordenar a la computadora realizar alguna acción o para contestar una pregunta. Los botones típicos son OK, Apply, Cancel, Close, Yes, No y Help.

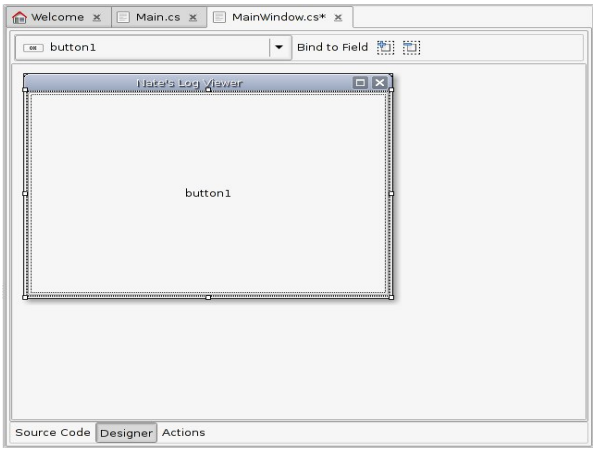

Fig. 3 Creación de button

Label: Permite el despliegue de texto o de una imagen. Ninguna funcionalidad de interacción de usuario se proporciona por defecto, pero sí se puede derivar para crear un nuevo widget que interaccione con el usuario.

CheckBox: Es un Checkbox con una etiqueta de texto. CheckBox es un botón de opción, es decir ellos pueden activarse o desactivarse, las cajas de CheckBox definen muchas de muchas opciones. En un grupo de botones (ButtonGroup), en un momento puede verificarse, si el usuario selecciona otro botón, el botón previamente seleccionado queda activo, ya que este se puede seleccionar varios a la misma vez.

ComboBox: Es la combinación de un botón y una lista popup. Muestra el elemento actual seleccionado de la lista y puede desplegar la lista de los elementos con un menú pop up.

MenuBar: Es un widget considerado para generar menús desplegables.

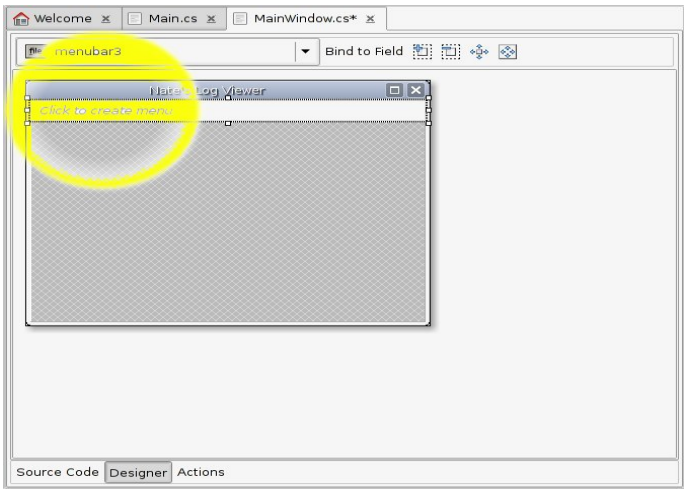

Fig. 4 Diseño de menús

Vbox: Este Widget es un contenedor y se utiliza para la ubicación de los componentes del formulario.

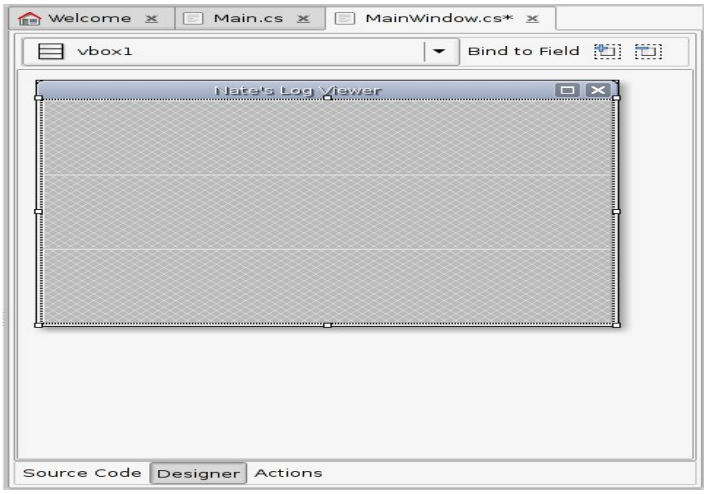

Fig. 5 Diseño de Vbox

5.4 MySQL

Se maneja que para acceder a cualquier base de datos es mucho más útil usar un motor o servidor que hace las funciones de intérprete entre las aplicaciones y usuarios con las bases de datos.

Esta utilidad se traduce en ventajas, entre las que podemos mencionar:

- Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
- Seguridad, en forma de permisos y privilegios, determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas. Esto permite compartir datos sin que peligre la integridad de la base de datos o proteja determinados contenidos.
- Potencia: SQL es un lenguaje muy potente para consulta de bases de datos, usar un motor nos ahorra una enorme cantidad de trabajo.
- Portabilidad: SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas. Esto, unido al uso de C/C++/C# proporciona una portabilidad enorme.

  En concreto, usar MySql tiene ventajas adicionales:

  - Escalabilidad: es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
  - MySQL está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes.
  - Conectividad: es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.

- Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador.
- Permite manejar multitud de tipos para columnas.
- Permite manejar registros de longitud fija o variable.

5.4.1 Definiciones

¿Qué son las bases de datos? Vamos a explicar unas bases generales, aunque sólidas, suficientes para desarrollar la mayor parte de los pequeños y medianos proyectos. Hay que señalar que existen varios modelos lógicos de bases de datos, aunque sólo veremos el modelo de ***bases de datos relacionales.***

Dato: Podemos decir que un dato es una información que refleja el valor de una característica de un objeto real, sea concreto, abstracto, o imaginario. Debe cumplir algunas condiciones, por ejemplo, debe permanecer en el tiempo. En ese sentido, estrictamente hablando, una edad no es un dato, ya que varía con el tiempo. El dato sería la fecha de nacimiento, y la edad se calcula a partir de ese dato y de la fecha actual. Además, debe tener un significado, y debe ser manipulable mediante operadores: comparaciones, sumas, restas, etc. (por supuesto, no todos los datos admiten todos los operadores).

Base de datos: Podemos considerar que es un conjunto de datos de varios tipos, organizados e interrelacionados. Estos datos deben estar libres de redundancias innecesarias y ser independientes de los programas que los usan.

SGBD (DBMS): Son las siglas que significan Sistema de Gestión de Bases de Datos, en inglés DBMS, Data Base Manager System. En este caso, MySQL es un SGBD, o mejor dicho: nuestro SGBD.

Consulta: Es una petición al SGBD para que procese un determinado comando SQL. Esto incluye tanto peticiones de datos como creación de bases de datos, tablas, modificaciones, inserciones, etc.

Redundancia de datos: Hay redundancia de datos cuando la misma información es almacenada varias veces en la misma base de datos. Esto es siempre algo a evitar, la redundancia dificulta la tarea de modificación de datos, y es el motivo más frecuente de inconsistencia de datos. Además, requiere un mayor espacio de almacenamiento, que influye en mayor coste y mayor tiempo de acceso a los datos.

Inconsistencia de datos: Sólo se produce cuando existe redundancia de datos. La inconsistencia se refiere a que no todas las copias redundantes contienen la misma información. Así, si existen diferentes modos de obtener la misma información, y esas formas pueden conducir a datos almacenados en distintos sitios. El problema surge al modificar esa información, si sólo cambiamos esos valores en algunos de los lugares en que se guardan, las consultas que hagamos más tarde podrán dar como resultado respuestas inconsistentes (es decir, diferentes). Puede darse el caso de que dos aplicaciones diferentes proporcionen resultados distintos para el mismo dato.

Integridad de datos: Cuando se trabaja con bases de datos, generalmente los datos se reparten entre varios ficheros. Así como pasa con MySql, la base de datos está disponible para varios usuarios de forma simultánea, deben existir mecanismos que aseguren que las interrelaciones entre registros se mantengan coherentes, es decir, que se respetan las dependencias de existencia y que las claves únicas no se repitan.

5.4.2 Diseño de bases de datos: El Modelo conceptual y El modelo Entidad-Relación.

Siempre que emprendamos un nuevo proyecto, grande o pequeño, antes de lanzarnos a escribir código, crear tablas o bases de datos, hay que analizar el problema sobre el papel. En el caso de las bases de datos, pensaremos sobre qué tipo de información necesitamos guardar, o lo que es más importante: qué tipo de

información necesitaremos obtener de la base de datos. En esto consiste el modelado de bases de datos.

Modelado de bases de datos

El proceso de trasladar un problema del mundo real a un ordenador, usando bases de datos, se denomina modelado. Para el modelado de bases de datos es necesario seguir un procedimiento determinado, seguir todo el proceso nos facilitará una documentación necesaria para revisar o mantener la aplicación, ya sea por nosotros mismos o por otros administradores o programadores.

La primera fase del diseño de una aplicación (la base de datos, generalmente, es parte de una aplicación), consiste en hablar con el cliente para saber qué quiere, y qué necesita realmente. Los modelos conceptuales ayudan en esta fase del proyecto, ya que facilitan una forma clara de ver el proceso en su totalidad, puesto que se trata de una representación gráfica.

Los modelos conceptuales no están orientados a ningún sistema físico concreto: tipo de ordenador, sistema operativo, SGBD, etc. Además de consultar con el cliente, una buena técnica consiste en observar el funcionamiento del proceso que se quiere informatizar o modelar. Con las bases de datos lo más importante es observar qué tipo de información se necesita, y que parte de ella se necesita con mayor frecuencia. Una vez recogidos los datos, el siguiente paso es crear un modelo conceptual. El modelo más usado en bases de datos es el

La siguiente fase es convertir el modelo conceptual en un modelo lógico. Existen varios modelos lógicos, pero el más usado, el que mejor se adapta a **MySQL** y el que por lo tanto el que utilizaremos, es el **modelo Relacional**.

La conversión entre el modelo conceptual y el lógico es algo bastante mecánico, aunque no por ello será siempre sencillo. El modelo lógico relacional, consta de un proceso que sirve para verificar que hemos aplicado bien el modelo, y en caso

contrario, corregirlo para que sea así. Este proceso se llama normalización, y también es bastante mecánico.

La última fase consiste en codificar el modelo lógico en un modelo físico. Este proceso está ligado al DBMS elegido, en el caso del DBMS que ocuparemos es MySQL, que vamos a manejar, para esto se requiere el conocimiento del lenguaje de consulta SQL.

Modelo Entidad-Relación

En esencia, el modelo entidad-relación (en adelante E-R), consiste en buscar las entidades que describan los objetos que intervienen en el problema y las relaciones entre esas entidades. Todo esto se plasma en un esquema gráfico que tiene por objeto, por una parte, ayudar al programador durante la codificación, y por otra, al usuario a comprender el problema y el funcionamiento del programa.

Entidad: Estamos hablando del modelo Entidad-Relación, por lo tanto este es un concepto que no podemos dejar sin definir. Es una representación de un objeto individual concreto del mundo real.

Conjunto de entidades: Es la clase o tipo al que pertenecen entidades con características comunes.

Atributo: Es cada una de las características que posee una entidad, y que agrupadas permiten distinguirla de otras entidades del mismo conjunto.

Dominio: Es conjunto de valores posibles para un atributo.

Relación: El otro concepto que no podemos dejar de definir es el de relación. Aunque en realidad, salvo para nombrar el modelo, usaremos el término interrelación, ya que relación tiene un significado radicalmente diferente dentro del modelo relacional, y esto nos puede llevar a error.

Interrelación: Es la asociación o conexión entre conjuntos de entidades.

Grado: Es el número de conjuntos de entidades que intervienen en una interrelación, existen interrelaciones de grado 2, 3, 4, etc. Pero las más frecuentes son las interrelaciones binarias.

Clave: Estaremos de acuerdo en que es muy importante poder identificar claramente cada entidad y cada interrelación. Esto es necesario para poder referirnos a cada elemento de un conjunto de entidades o interrelaciones, ya sea para consultarlo, modificarlo o borrarlo. No deben existir ambigüedades en ese sentido. Es un conjunto de atributos que identifican de forma unívoca una entidad.

Claves candidatas: Una característica que debemos buscar siempre en las claves es que contengan el número mínimo de atributos, siempre que mantengan su función. Diremos que una clave es mínima cuando si se elimina cualquiera de los atributos que la componen, deja de ser clave.

Si en una entidad existe más de una de estas claves mínimas, cada una de ellas es una clave candidata**.** Es cada una de las claves mínimas existente en un conjunto de entidades.

Clave principal: Si disponemos de varias claves candidatas no usaremos cualquiera de ellas según la ocasión. Esto sería fuente de errores, de modo que siempre usaremos la misma clave candidata para identificar la entidad. Es una clave candidata elegida de forma arbitraria, que usaremos siempre para identificar una entidad.

Claves de interrelaciones: Para identificar interrelaciones el proceso es similar, aunque más simple. Tengamos en cuenta que para definir una interrelación usaremos las claves primarias de las entidades interrelacionadas.

Entidades fuertes y débiles: Las entidades débiles no necesitan una clave primaria, sus claves siempre están formadas como la combinación de una clave primaria de una entidad fuerte y otros atributos y las entidades fuertes siempre tienen su clave primaria.

5.4.3 Representación de Entidades y Relaciones:

Diagramas: Un diagrama E-R consiste en representar mediante figuras un modelo completo del problema, proceso o realidad a describir, de forma que se definan tanto las entidades que lo componen, como las interrelaciones que existen entre ellas.

Entidad: Las entidades se representan con un rectángulo, y en su interior el nombre de la entidad:

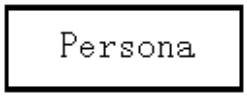

Fig. 6 Entidad

Atributo: Los atributos se representan mediante elipses, y en su interior el nombre del atributo:

Fig. 7 Atributo

Algunas variantes de diagramas E-R usan algunas marcas para indicar que cierto atributo es una clave primaria, como subrayar el nombre del atributo.

Fig. 8 Atributo2

También es frecuente usar una doble elipse para indicar atributos multivaluados:

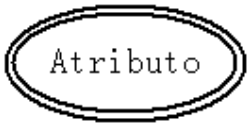

Fig. 9 Atributo multivaluado

Atributo multivaluado: (o multivalorado) se dice del atributo tal que para una misma entidad puede tomar varios valores diferentes, es decir, varios valores del mismo dominio.

Interrelación: Las interrelaciones se representan mediante rombos, y en su interior el nombre de la interrelación:

Fig.10 Interrelación

En los extremos de las líneas que parten del rombo se añaden unos números que indican la cantidad de entidades que intervienen en la interrelación: **1, n**. Esto también se suele hacer modificando el extremo de las líneas. Si terminan con un extremo involucran a una entidad, si terminan en varios extremos, (generalmente tres), involucrarán a varias entidades:

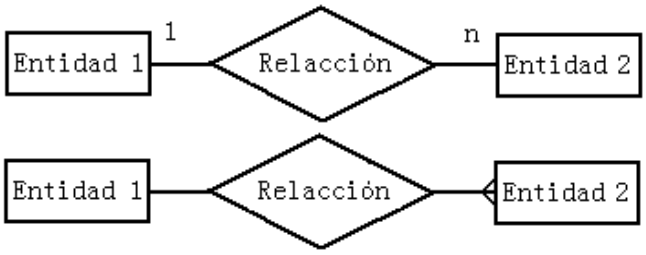

Fig. 11 Interrelación 2

Sobre las líneas a veces se añade el rol que representa cada entidad.

Dominio: A veces es conveniente añadir información sobre el dominio de un atributo, los dominios se representan mediante hexágonos, con la descripción del dominio en su interior.

Fig. 12 Dominio

5.4.4 Diseño de bases de datos: El modelo lógico y El modelo relacional

Modelo relacional: Entre los modelos lógicos, el modelo relacional está considerado como el más simple. El paso del modelo E-R al relacional es muy simple, y por otra, MySQL, como implementación de SQL, está orientado principalmente a bases de datos relacionales.

El modelo se compone de tres partes:

- **Estructura de datos:** básicamente se compone de relaciones.
- **Manipulación de datos:** un conjunto de operadores para recuperar, derivar o modificar los datos almacenados.
- **Integridad de datos:** una colección de reglas que definen la consistencia de la base de datos.

Relación: Es un conjunto de datos referentes a un conjunto de entidades y organizados en forma tabular, que se compone de filas y columnas, (tuplas y atributos), en la que cada intersección de fila y columna contiene un valor.

Tupla: Cada una de las filas de una relación. Contiene la información relativa a una única entidad. De esta definición se deduce que no pueden existir dos tuplas iguales en la misma relación.

Nulo (NULL): Valor asignado a un atributo que indica que no contiene ninguno de los valores del dominio de dicho atributo.

Modelo relacional: Es un modelo basado en relaciones, en la que cada una de ellas cumple determinadas condiciones mínimas de diseño:

1. No deben existir dos tuplas iguales.
2. Cada atributo sólo puede tomar un único valor del dominio, es decir, no pueden contener listas de valores.
3. El orden de las tuplas dentro de la relación y el de los atributos, dentro de cada tupla, no es importante**.**

Cardinalidad: Número de tuplas que contiene una relación.

Esquema: Es la parte constante de una relación, es decir, su estructura. Esto es, el esquema de una lista de los atributos que definen una relación y sus dominios.

Instancia: Es el conjunto de las tuplas que contiene una relación en un momento determinado. Es como una fotografía de la relación, que sólo es válida durante un periodo de tiempo concreto.

Clave: Es un conjunto de atributos que identifica de forma unívoca a una tupla. Puede estar compuesto por un único atributo o una combinación de varios.
Podemos clasificar las claves en distintos tipos:

- **Candidata:** cada una de las posibles claves de una relación, en toda relación existirá al menos una clave candidata. Esto implica que ninguna relación puede contener tuplas repetidas.
- **Primaria:** (o principal) es la clave candidata elegida por el usuario para identificar las tuplas. No existe la necesidad, desde el punto de vista de la teoría de bases de datos relacionales, de elegir una clave primaria. Además, las claves primarias no pueden tomar valores nulos. Es preferible por motivos de optimización de MySQL, que estos valores sean enteros, aunque no es obligatorio. MySQL sólo admite una clave primaria por tabla, que es lógico, ya que la definición implica que sólo puede existir una.

- **Alternativa:** cada una de las claves candidatas que no son clave primaria, si es que existen.
- **Foránea:** (o externa) es el atributo (o conjunto de atributos) dentro de una relación que contienen claves primarias de otra relación. No hay nada que impida que ambas relaciones sean la misma.

Interrelación: Dos relaciones están interrelacionadas cuando una posee una clave foránea de la otra.

Según esto, existen dos tipos de interrelación:

- La interrelación entre entidades fuertes y débiles.
- La interrelación pura, entre entidades fuertes.

Al igual que en el modelo E-R, existen varios tipos de interrelación:

- **Uno a uno:** a cada tupla de una relación le corresponde una y sólo una tupla de otra.
- **Uno a varios:** a cada tupla una relación le corresponden varias en otra.
- **Varios a varios:** cuando varias tuplas de una relación se pueden corresponder con varias tuplas en otra.

5.4.5 Paso del modelo E-R al modelo relacional

Existen varias reglas para convertir cada uno de los elementos de los diagramas E-R en tablas:

Para cada conjunto de entidades fuertes se crea una relación con una columna para cada atributo.

Para cada conjunto de entidades débiles se crea una relación que contiene una columna para los atributos que forman la clave primaria de la entidad fuerte a la que se encuentra subordinada y una columna para cada atributo de la entidad.

Para cada interrelación se crea una relación que contiene una columna para cada atributo correspondiente a las claves principales de las entidades interrelacionadas.

Lo mismo para entidades compuestas, añadiendo las columnas necesarias para los atributos añadidos a la interrelación.

Las relaciones se representan mediante sus esquemas, la sintaxis es simple: <nombre_relación> (<nombre_atributo_i>,...) La clave principal se suele indicar mediante un subrayado.

5.4.6 Integridad de datos: El modelo relacional también provee mecanismos para mantener la integridad. Podemos dividir estos mecanismos en dos categorías:

- **Restricciones estáticas**, se refieren a los estados válidos de datos almacenados.
- **Restricciones dinámicas**, definen las acciones a realizar para evitar ciertos efectos secundarios no deseados cuando se realizan operaciones de modificación o borrado de datos.

5.4.7 Restricciones sobre claves primarias

En cuanto a las restricciones estáticas, las más importantes son las que afectan a las claves primarias. Ninguna de las partes que componen una clave primaria puede ser *NULL*. Las modificaciones de claves primarias deben estar muy bien controladas. Dado que una clave primaria identifica de forma unívoca a una tupla en una relación, parece poco lógico que exista necesidad de modificarla, ya que eso implicaría que no estamos definiendo la misma entidad.

Integridad referencial: La integridad referencial consiste en que si un atributo o conjunto de atributos se define como una clave foránea, sus valores deben existir en la tabla en que ese atributo es clave principal.

Existen varias formas de asegurarse de que se conserva la integridad referencial:

- **Restringir operaciones*:*** borrar o modificar tuplas cuya clave primaria es clave foránea en otras tuplas, sólo estará permitido si no existe ninguna tupla con ese valor de clave en ninguna otra relación. Es decir, si el valor de una clave primaria en una tupla es "clave1", sólo podremos eliminar esa tupla si el valor "clave1" no se usa en ninguna otra tupla, de la misma relación o de otra, como valor de clave foránea.

- **Transmisión en cascada:** borrar o modificar tuplas cuya clave primaria es clave foránea en otras implica borrar o modificar las tuplas con los mismos valores de clave foránea. Si en el caso anterior, modificamos el valor de clave primaria "clave1" por "clave2", todas las apariciones del valor "clave1" en donde sea clave foránea deben ser sustituidos por "clave2".

- **Poner a nulo*:*** cuando se elimine una tupla cuyo valor de clave primaria aparece en otras relaciones como clave foránea, se asigna el valor NULL a dichas claves foráneas. De nuevo, siguiendo el ejemplo anterior, si eliminamos la tupla con el valor de clave primaria "clave1", en todas las tuplas donde aparezca ese valor como clave foránea se sustituirá por NULL.

Tipos de columnas: En MySQL existen bastantes tipos diferentes disponibles, de modo que será mejor que los agrupemos por categorías: de caracteres, enteros, de coma flotante, tiempos, bloques, enumerados y conjuntos.

Tipos de datos de cadenas de caracteres utilizados

VARCHAR () [NATIONAL] VARCHAR (M) [BINARY] Contiene una cadena de longitud variable. Los valores válidos para M son de 0 a 255, y de 1 a 255 en versiones de MySQL anteriores a 4.0.2.Los espacios al final se eliminan. Si no se especifica la palabra clave BINARY estos valores se ordenan y comparan sin

distinguir mayúsculas y minúsculas. VARCHAR es un alias para CHARACTER VARYING.

Tipos de dato entero utilizado

INT

INT [(M)] [UNSIGNED] [ZEROFILL]

Contiene un entero de tamaño normal. El rango con signo está entre -2147483648 y 2147483647. El rango sin singo, entre 0 y 4294967295.

INTEGER

INTEGER [(M)] [UNSIGNED] [ZEROFILL]

Es sinónimo de INT.

DATE

TINIINT

Tipos de datos para datos sin tipo o grandes bloques de datos

MEDIUMBLOB
MEDIUMTEX

Contiene una columna BLOB o TEXT con una longitud máxima de 16777215 caracteres (2^{24})

LONGBLOB

LONGTEXT

LONGBLOB

LONGTEXT

Contiene una columna BLOB o TEXT con una longitud máxima de 4294967298 caracteres (2^{32})

5.4.8 Cliente MySQL

Existen muchas formas de establecer una comunicación con el servidor de MySql. En nuestros programas, generalmente, usaremos un API para realizar las consultas con el servidor. Usando una interfaz grafica como Monodevelop, por ejemplo, este API está integrado con el lenguaje, en C++ se trata de librerías de enlace dinámico, etc.

Para este trabajo monográfico usaremos MySql de forma directa, mediante un cliente ejecutándose en una consola (una ventana DOS en Windows, o un Shell en Linux u otros sistemas).

Veamos un ejemplo sencillo. Para ello abrimos una consola y tecleamos "MySql". (Si estamos en Windows y no está definido el camino para MySql tendremos que hacerlo desde "C:\mysql\bin").

Para entrar en la consola de MySql se requieren ciertos parámetros. Se debe tener presente que el servidor es multiusuario, y que cada usuario puede tener distintos privilegios, tanto de acceso a tablas como de comandos que puede utilizar. La forma general de iniciar una sesión MySql es:

mysql -h host -u usuario –p

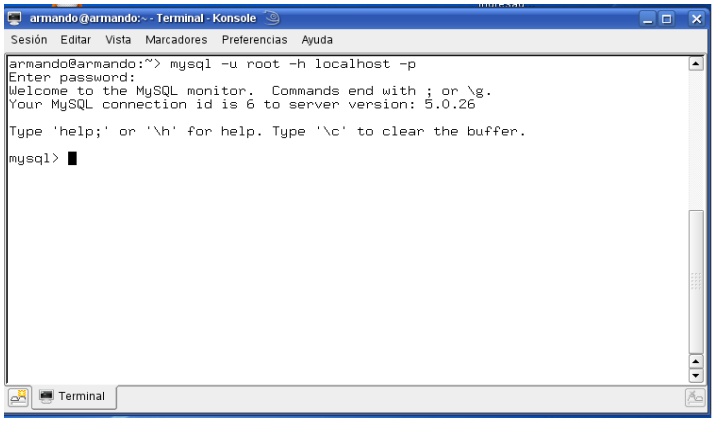

Fig.13 Terminal de arranque de MySQL

Podemos especificar el ordenador donde está el servidor de bases de datos (host) y nuestro nombre de usuario. Los parámetros "-h" y "-u" indican que los parámetros a continuación son, respectivamente, el nombre del host y el usuario. El parámetro "-p" indica que se debe solicitar una clave de acceso.

En versiones de MySQL anteriores a la 4.1.9 es posible abrir un cliente de forma anónima sin especificar una contraseña. Pero no es correcto, y de hecho, las últimas versiones de MySQL no lo permiten.

Durante la instalación de MySQL se nos pedirá que elijamos una clave de acceso para el usuario 'root', deberemos usar esa clave para iniciar una sesión con el cliente MySQL.

```
mysql -h localhost -u root -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 76 to server version: 4.1.9-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Para salir de una sesión del cliente de MySQL se usa el comando "QUIT".

```
mysql> QUIT
```

Bye y retornamos a la shell.

Algunas consultas

Para hacer consultas se usa la sentencia SQL SELECT, por ejemplo:

```
mysql> SELECT VERSION (), CURRENT_DATE;
+---------------------+------------------+
| VERSION () | CURRENT_DATE |
+---------------------+------------------+
| 5.0.26          |   2007-10-30       |
+---------------------+------------------+
1 row in set (0.50 sec)
mysql>
```

SELECT es la sentencia SQL para seleccionar datos de bases de datos, pero también se puede usar, como en este caso, para consultar variables del sistema o resultados de funciones. En este caso hemos consultado el resultado de la función VERSION y de la variable CURRENT_DATE.

Esto es sólo un aperitivo, hay muchísimas más opciones, y mucho más interesantes. Usaremos SELECT para mostrar todas las filas de una tabla. Para ello se usa la sentencia: mysql> SELECT * FROM <tabla>;

5.4.9 Usuarios y privilegios

Cuando se trabaja con bases de datos reales y con aplicaciones de gestión de bases de datos, es muy importante definir otros usuarios, además del root, que es el administrador. Esto nos permitirá asignar distintos privilegios a cada usuario, y nos ayudará a proteger las bases de datos.

Podremos por ejemplo, crear un usuario que sólo tenga posibilidad de consultar datos de determinadas tablas o bases de datos, pero que no tenga permiso para añadir o modificar datos, o modificar la estructura de la base de datos.
Otros usuarios podrán insertar datos, y sólo algunos (o mejor, sólo uno) podrán modificar la estructura de las bases de datos: los administradores.

5.4.10 Lenguaje SQL

Creación de bases de datos y tablas. A nivel teórico, existen dos lenguajes para el manejo de bases de datos:

DDL (Data Definition Language) Lenguaje de definición de datos. Es el lenguaje que se usa para crear bases de datos y tablas, y para modificar sus estructuras, así como los permisos y privilegios. Este lenguaje trabaja sobre unas tablas especiales llamadas diccionario de datos.

DML (Data Manipulation Language) lenguaje de manipulación de datos. Es el que se usa para modificar y obtener datos desde las bases de datos.

SQL engloba ambos lenguajes DDL+DML, ambos forman parte del conjunto de sentencias de SQL.

Crear una base de datos

Cada conjunto de relaciones que componen un modelo completo forma una base de datos. Desde el punto de vista de SQL, una base de datos es sólo un conjunto de relaciones (o tablas), y para organizarlas o distinguirlas se accede a ellas

mediante su nombre. A nivel de sistema operativo, cada base de datos se guarda en un directorio diferente.

Crear una base de datos es una tarea muy simple. Claro que, en el momento de crearla, la base de datos estará vacía, es decir, no contendrá ninguna tabla.

Para empezar, crearemos una base de datos solo para nosotros, y la llamaremos "Gimnasio". Para crear una base de datos se usa una sentencia.

CREATE DATABASE

```
mysql> CREATE DATABASE gimnasio;
Query OK, 1 row affected (0.04 sec)
```

Podemos averiguar cuántas bases de datos existen en nuestro sistema usando la sentencia SHOW DATABASES:

```
mysql> SHOW DATABASES;
+----------------------------------+
| Database                         |
+----------------------------------+
| information_schema               |
|gimnasio                          |
| mysql                            |
| test                             |
+----------------------------------+
4 rows in set (0.31 sec)
```

Seleccionaremos Database como la base de datos por defecto. Esto nos permitirá obviar el nombre de la base de datos en consultas. Para seleccionar una base de datos se usa el comando **USE**, que no es exactamente una sentencia SQL, sino más bien de una opción de MySQL:

```
mysql> USE gimnasio;
Database changed
mysql>
```

Crear una tabla

La sentencia CREATE TABLE que sirve para crear tablas. La sintaxis de esta sentencia es muy compleja, ya que existen muchas opciones y tenemos muchas posibilidades diferentes a la hora de crear una tabla.

La sentencia CREATE TABLE creará una tabla con las columnas que indiquemos. Crearemos una tabla que nos permitirá almacenar nombres de usuario y su contraseña. Deberemos indicar el nombre de la tabla y los nombres y tipos de las columnas:

```
mysql> USE gimnasio;
Database changed
mysql> CREATE TABLE usuarios (nombre VARCHAR (50), pass VARCHAR(50), tipo tinyint(1));
Query OK, 0 rows affected (0.53 sec)
mysql>
```

Hemos creado una tabla llamada "usuarios" con cuatro columnas: "nombre" que puede contener cadenas de hasta 50 caracteres, “pass” de hasta 50 caracteres de tipo varchar, “tipo” especifica si es usuario o administrador y “foto” de tipo longblob.

Podemos consultar cuántas tablas y qué nombres tienen en una base de datos, usando la sentencia SHOW TABLES:

```
mysql> SHOW TABLES;
+-------------------------+
| Tables_in_gimnasio |
+-------------------------+
| usuarios                |
+-------------------------+
1 row in set (0.01 sec)
mysql>
```

La sintaxis para definir columnas es:

nombre_col tipo [NOT NULL | NULL] [DEFAULT valor_por_defecto]
[AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
[definición_referencia]

Opciones de columna

Valores nulos: Al definir cada columna podemos decidir si podrá o no contener valores nulos. Aquellas columnas que son o forman parte de una clave primaria no pueden contener valores nulos.

Columna auto incrementadas

En MySQL tenemos la posibilidad de crear una columna auto incrementada, aunque esta columna sólo puede ser de tipo entero. Si al insertar una fila se omite el valor de la columna auto incrementado o si se inserta un valor nulo para esa columna, su valor se calcula automáticamente, tomando el valor más alto de esa columna y sumándole una unidad. Esto permite crear, de una forma sencilla, una columna con un valor único para cada fila de la tabla.

Comentarios

Adicionalmente, al crear la tabla, podemos añadir un comentario a cada columna. Este comentario sirve como información adicional sobre alguna característica especial de la columna, y entra en el apartado de documentación de la base de datos: usamos

Comillas simples

Como creamos las definición de las tablas

CREATE TABLE usuarios (nombre VARCHAR (50), pass VARCHAR (50));

5.4.10.1 Inserción y modificación de datos

Una base de datos sin datos no sirve para mucho, de modo que veremos cómo agregar, modificar o eliminar los datos que contiene nuestra base de datos.

Inserción de nuevas filas

La forma más directa de insertar una fila nueva en una tabla es mediante una sentencia INSERT. En la forma más simple de esta sentencia debemos indicar la tabla a la que queremos añadir filas, y los valores de cada columna. Las columnas de tipo cadena o fechas deben estar entre comillas sencillas o dobles, para las columnas numéricas esto no es imprescindible, aunque también pueden estar entrecomilladas.

Reemplazar filas

Existe una sentencia REPLACE, que es una alternativa para INSERT, que sólo se diferencia en que si existe algún registro anterior con el mismo valor para una clave primaria o única, se elimina el viejo y se inserta el nuevo en su lugar.

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [(col_name...)]
VALUES ({expr | DEFAULT}...), (...)...
REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name
SET col_name= {expr | DEFAULT}...

Actualizar filas

Podemos modificar valores de las filas de una tabla usando la sentencia **UPDATE**. En su forma más simple, los cambios se aplican a todas las filas, y a las columnas que especifiquemos.

UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]

Eliminar filas

Para eliminar filas se usa la sentencia **DELETE**. La sintaxis es muy parecida a la de **UPDATE**:

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM table_name
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]

La forma más simple es no usar ninguna de las cláusulas opcionales.

Vaciar una tabla

Cuando queremos eliminar todas las filas de una tabla, vimos en el punto anterior que podíamos usar una sentencia DELETE sin condiciones. Sin embargo, existe una sentencia alternativa, TRUNCATE, que realiza la misma tarea de una forma mucho más rápida.

La diferencia es que DELETE hace un borrado secuencial de la tabla, fila a fila. Pero TRUNCATE borra la tabla y la vuelve a crear vacía, lo que es mucho más eficiente

Selección de datos

Usaremos las Sentencias SELECT
Sintaxis:

SELECT [ALL | DISTINCT | DISTINCTROW]
expresion_select...
FROM referencias_de_tablas
WHERE condiciones
[GROUP BY {nombre_col | expresion | position}
[ASC | DESC]... [WITH ROLLUP]]
[HAVING condiciones]
[ORDER BY {nombre_col | expresión | posición}
[ASC | DESC],...]
[LIMIT {[desplazamiento,] contador | contador OFFSET desplazamiento}]

Forma incondicional

La forma más sencilla es la que hemos usado hasta ahora, consiste en pedir todas las columnas y no especificar condiciones.

mysql> SELECT * FROM usuarios;

Limitar las columnas: proyección

Una de las operaciones del álgebra relacional era la proyección, que consistía en seleccionar determinados atributos de una relación, mediante la sentencia SELECT es posible hacer una proyección de una tabla, seleccionando las columnas de las que queremos obtener datos.

Limitar las filas: selección

Otra de las operaciones del álgebra relacional era la selección, que consistía en seleccionar filas de una relación que cumplieran determinadas condiciones.

Agrupar filas

Es posible agrupar filas en la salida de una sentencia SELECT según los distintos valores de una columna, usando la cláusula GROUP BY. Esto, en principio, puede parecer redundante, ya que podíamos hacer lo mismo usando la opción DISTINCT. Sin embargo, la cláusula GROUP BY es más potente:

Ordenar resultados

Podemos añadir una cláusula de orden ORDER BY para obtener resultados ordenados por la columna que queramos. Existe una opción para esta cláusula para elegir el orden, ascendente o descendente. Se puede añadir a continuación ASC o DESC, respectivamente. Por defecto se usa el orden ascendente, de modo que el modificador ASC es opcional.

VI. Metodología del documento

- Recopilación de la información: Toda la información que se necesitó para la realización de este trabajo monográfico, se obtuvo mediante búsquedas en Internet, libros y manuales.

- Especificación de requisitos: Los requisitos estarán enfocados a nivel de hardware y software.

- Estándares: La aplicación de los estándares en la creación de la aplicación, nos garantiza la reusabilidad y migración entre instalaciones o plataformas diferentes.

    - Tipo de licencia de uso: Utilizamos licencia GPL (Global Public License). En éste punto se ha tenido en cuenta las ventajas de adaptación que ofrece el software libre, ya que la plataforma debe permitir la integración con nuestros sistemas de información, así como su personalización para una correcta integración.

    - Seguridad: El gestor MySQL nos provee una excelente seguridad de los datos.

    - Interactividad: El usuario interactúa con el sistema introduciendo datos en el y obteniendo a su vez resultados.

- Plataforma a utilizar: Hemos elegido Linux openSUSE 10.3 debido a su facilidad de uso, además, es un software libre respaldado por Novell.

- Instalación del sistema Linux openSUSE 10.3: Teniendo la plataforma sobre la cual trabajaremos, será necesaria la instalación, configuración del sistema operativo Linux openSUSE 10.3, porque es un sistema fiable, seguro, y principalmente de distribución libre.

- Instalación y configuración de MySQL: Una vez instalada la plataforma seleccionada procedemos a la instalación del servidor y cliente MySQL, para ello haremos uso de la herramienta gráfica YaST.

- Instalación y configuración del creador de interfaces gráficas Monodevelop: Luego de haber instalado MySQL procedemos a instalar el creador de interfaces gráficas Monodevelop utilizando también la herramienta Yast.

- Personalización: Se procedió a personalizar la interfaz gráfica de la plataforma para que facilite el acceso a la información, presentando al usuario final una serie de herramientas básicas y útiles para una mejor interacción.

- Explotación y Mantenimiento: Se hará una prueba experimental con la Aplicación *"Registro y Control de clientes del gimnasio Acción Club"*, para demostrar la utilidad y facilidad de uso del creador de interfaces gráficas Monodevelop y la interoperatividad con MySQL, y resolver las dificultades que se presenten durante el periodo de prueba.

VII. Metodología de la Aplicación

El método de desarrollo que se ha seleccionado para la elaboración de la Aplicación: “Registro y Control de clientes del Gimnasio Acción Club” bajo entorno Linux, es el Método del Ciclo de Vida Clásico o en Cascada (Ver Fig. 14), ya que resulta conveniente en la identificación de las actividades o etapas a seguir en la elaboración de la Aplicación.

Las etapas a desarrollar durante el proceso investigativo son:

- Investigación preliminar o Ingeniería del Sistema.
- Análisis de Requisitos del Software.
- Diseño del Sistema.
- Codificación.
- Prueba.
- Implementación y evaluación.

Cada una de estas etapas lleva asociada una serie de tareas que deberán realizarse y una serie de documentos que serán las salidas de cada una de estas fases y que servirán de entrada a la siguiente fase, de esta manera se logra progresar a través del análisis, diseño, codificación, prueba e implementación, sin llegar a la etapa de mantenimiento, pues la Aplicación no será instalada en una empresa, que requiera darle mantenimiento posterior.

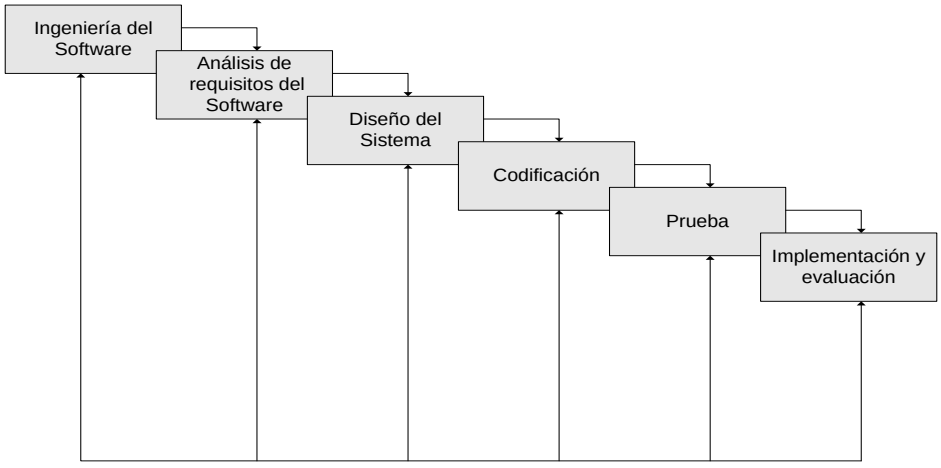

Fig. 14 Modelo de vida en Cascada

- Investigación preliminar o Ingeniería del Sistema: En esta fase se realiza el estudio de factibilidad técnica, económica y operacional de la Aplicación.

- Análisis de Requisitos del Software: Se debe comprender la funcionalidad que tendrá la Aplicación, así como su rendimiento y la interfaz requerida.

- Diseño del Sistema: Se diseñan procedimientos precisos para especificar la interfaz que utilizará la Aplicación, tomando como herramienta el Diseñador de interfaces (Monodevelop) y el IDE para desarrollo en GNOME(C#),

- Codificación: El diseño se traduce de forma precisa a la computadora, utilizando un lenguaje de programación de alto nivel, en nuestro caso C#. además de generar la interfaz gráfica de la Aplicación.

- Prueba: Una vez generado el código respectivo, comienza la fase de prueba de la Aplicación. Confirmando que se puede llevar a cabo el registro y control de clientes a través de la aplicación principal.

- Implementación y evaluación: Una vez que el código de los programas ha pasado la fase de prueba y se han hecho las correcciones en caso de ser necesario, se procede a la instalación de las Aplicaciones propuestas. Debido a que la Aplicaciones no están diseñadas para implementarse en una compañía o empresa, su evaluación consistirá con en el uso de Aplicación y del presente trabajo monográfico por un número determinado de usuarios.

VIII. Recursos Disponibles

Los recursos mínimos necesarios para el desarrollo de nuestro trabajo son:

Recursos Hardware:

- Procesador: Intel Pentium IV de 1.8 GHz.
- Disco Duro: Maxtor 80 GB.
- Memoria Principal: ACE 512 MB, DDR.
- Unidad de CD/ROM, DVD/R, de 16x.
- 4 puertos USB.

Recursos Software:

a. Sistema Operativo OpenSuse Linux 10.3. GNOME.

b. Diseñador de interfaces Graficas (Monodevelop). Versión 0.15.

c. Gestor de Base de Datos MySQL 5.0.26

d. MySQL Administrator 5.0,

e. MySQL Query Browser 1.2.12

f. Herramienta utilizada para la redacción del informe final OpenOffice

g. Diseñadores Gráficos para imágenes y diagramas El GIMP y DIA

h. Navegadores Web utilizados para la recopilación de información: Firefox.

i. Driver MySQL Connector/.NET

IX. Análisis de Requisitos

9.1 Planteamiento de aplicación a resolver

Para la aplicación de nuestro trabajo elegimos el gimnasio “Acción Club” que necesita la implantación de una base de datos que le permita el registro y control de sus clientes, recursos utilizados, etc. Para ello nos facilito información de la infraestructura, medios, organización y forma de trabajo que se tradujeron en las especificaciones siguientes:

- Existen varias salas de las cuales se quiere guardar información como el tipo de sala (pesas y Aeróbicos), código de sala y capacidad de la sala.
- Hay salas que poseen aparatos y hay otras que no, en ellas se puede o no impartir clases de las diferentes modalidades (Pesas, Aeróbicos).
- Cada aparato esta asignado a una sala que tiene al igual que el aparato, un código que registra la vida útil del aparato.
- También, posee información de las clases que se imparten (código de la clase, hora, nombre del instructor y código de la sala en la que se imparte); cada clase se identifica por su código de clase. Cada clase tiene asignada una sala en la que se imparte, lo mismo que un instructor.
- De cada instructor se tiene información como identificación, nombre, apellido, edad y sexo, así como las clases que pueden impartir.
- De cada cliente se tiene información como su nombre, apellido, número de cliente, sexo, edad y teléfono.

Toda la información anterior fue necesaria para la utilización de la base de datos que creamos en nuestro trabajo monográfico.

9.2 Especificación de Requisitos Software del "Registro y Control de los clientes del gimnasio Acción Club"

1. Introducción

1.1 Propósito

Definición del conjunto de especificaciones de requisitos software que debe cumplir la aplicación de "Registro y control de los clientes del gimnasio Acción Club", consiste en la mecanización de los procesos de registros y servicios a los clientes.

Este documento se dirige a la dirección del gimnasio y a los usuarios finales quienes deberán estudiarlo para su aprobación o desacuerdo antes de abordar su fase de análisis

1.2 Alcance

El nombre con el que se conocerá esta aplicación es "Registro y Control de los clientes del gimnasio Acción Club".

El producto realizara las siguientes funciones:

- Registro de clientes.
- Registro de instructores.
- Control de clases impartidas por instructores.
- Anotación de monto de las clases.
- Control de las salas existentes en el gimnasio.
- Control. de aparatos existentes en las distintas salas.
- Emisión de listados de estudiantes por clase.
- Emisión de listados de los instructores de gimnasio.

1.3 Definiciones, Acrónimos y Abreviaturas

1.3.1. Instructor: persona

1.3.2. Cliente: Persona que paga por los servicios del gimnasio

1.3.3. Clase: Servicio brindado al cliente por parte del gimnasio

1.3.4. Sala: Lugar donde se desarrolla la clase.

1.3.5. Aparato: Instrumento que utiliza el cliente en las diferentes sesiones de clases.

1.4 Referencias

1.4.1. No existe referencia de un sistema automatizado para esta aplicación, solamente cuenta con registros en libros.

1.5 Visión General

1.5.1. Primeramente se realizó una descripción general del producto implementado y posteriormente se procedió a estudiar cada uno de los requisitos específicos individualmente.

2. Descripción General

2.1 Realizaciones del producto

La aplicación interactúa con la base de datos de registro de datos del gimnasio y también registros de cliente e instructores.

El equipo con que se desarrollo la aplicación posee las características siguientes:

Intel Pentium IV de 1.8 GHz.

RAM 512 MB

Disco duro 80 GB.

2.2 Funciones del producto

El producto software contiene todas las tareas que realiza el personal del gimnasio “Acción Club” de forma diaria:

- Registros de los clientes.

- Control de los Instructores.
- Registros de nuevos clientes.
- Control de las actualizaciones del registro de los clientes.
- Controlar el horario de las clases.
- Llevar registro de la vida útil de los aparatos.
- Control de salas y/o aulas.

2.3 Características del usuario

Los usuarios finales de la aplicación serán personas con experiencia en informática básica, lo que deberá incluir una manual de ayuda.

2.4 Restricciones Generales

El lenguaje de programación utilizado fue C# y se siguieron los estándares de la programación estructurada

3. Requisitos específicos

3.1 Ingreso de un nuevo cliente

Este proceso deberá realizar la captura de un nuevo cliente, todos los datos y las clases en las que se inscribirá el cliente mediante un formulario.

3.1.2 Entrada

Por pantalla: datos para ingresar al cliente

- Código del cliente.
- Nombre del cliente.
- Apellidos del cliente.
- Sexo del cliente.
- Edad del cliente.
- Teléfono del cliente.

3.2 Proceso

Se mostrará el formulario de introducción del usuario. Esté deberá tener un código (número) con el que se registrará el nuevo cliente de manera correlativa.

Los datos necesarios a introducir serán:

- Código del cliente: es un dato obligatorio. Debe existir en el fichero maestro de cliente o se indicará un error.
- Nombre del cliente: es un dato obligatorio. Debe existir en el fichero maestro de cliente o se indicará un error.
- Apellidos del cliente: es un dato obligatorio. Debe existir en el fichero maestro de cliente o se indicará un error.
- Sexo del cliente: dato optativo masculino o femenino
- Edad del cliente: dato optativo
- Teléfono del cliente: dato optativo

3.2.1 Salidas

Con todos los datos mencionados se almacenará el pedido en la base de datos del sistema y se registrara la entrada correspondiente

3.3 Interfaces Externas

3.3.1 Interfaces de usuario

La captura de datos del pedido se realizara de forma interactiva por pantalla.

3.3.2 Interfaces Hardware

Se podrá utilizar cualquier Terminal conectada al ordenador central.

3.3.3 Interfaces Software

El proceso interactúa con la base de datos de ficheros maestros ya mencionados.

3.3.4 Interfaces de comunicaciones

No existe ninguna interfaz de comunicaciones en la aplicación

3.4 Requisitos de funcionamiento

Requisitos estáticos: no existe ninguna restricción sobre el número de terminales y de usuarios que estén trabajando simultáneamente con el sistema.

Requisitos dinámicos: es importante que el tiempo de respuesta no aumente exponencialmente con el número de usuarios.

3.5 Restricciones de diseño

El formato de pantalla y listados de la aplicación deberá contener información acerca del nombre del gimnasio y el nombre del usuario.

3.6 Atributos

3.6.1 Seguridad

Todos los programas de la aplicación deberán estar protegidos mediante autorizaciones de uso.

3.6.2 Mantenimiento

Cualquier modificación que afecte a los requisitos mencionados en este documento, deberá ser reflejada el mismo, así como la documentación obtenida en la fase de análisis, diseño y programación.

3.7 Otros requisitos

3.7.1 base de datos

EL almacenamiento de información se realizará por medio de una base dato relacional.

3.7.2 Operaciones

Todas las operaciones sobre la base de datos se realizarán según lo mencionado en sub-apartado de seguridad.

X. Fase de Diseño

Modelo Conceptual de la aplicación “Registro y Control de Clientes del gimnasio Acción Club”

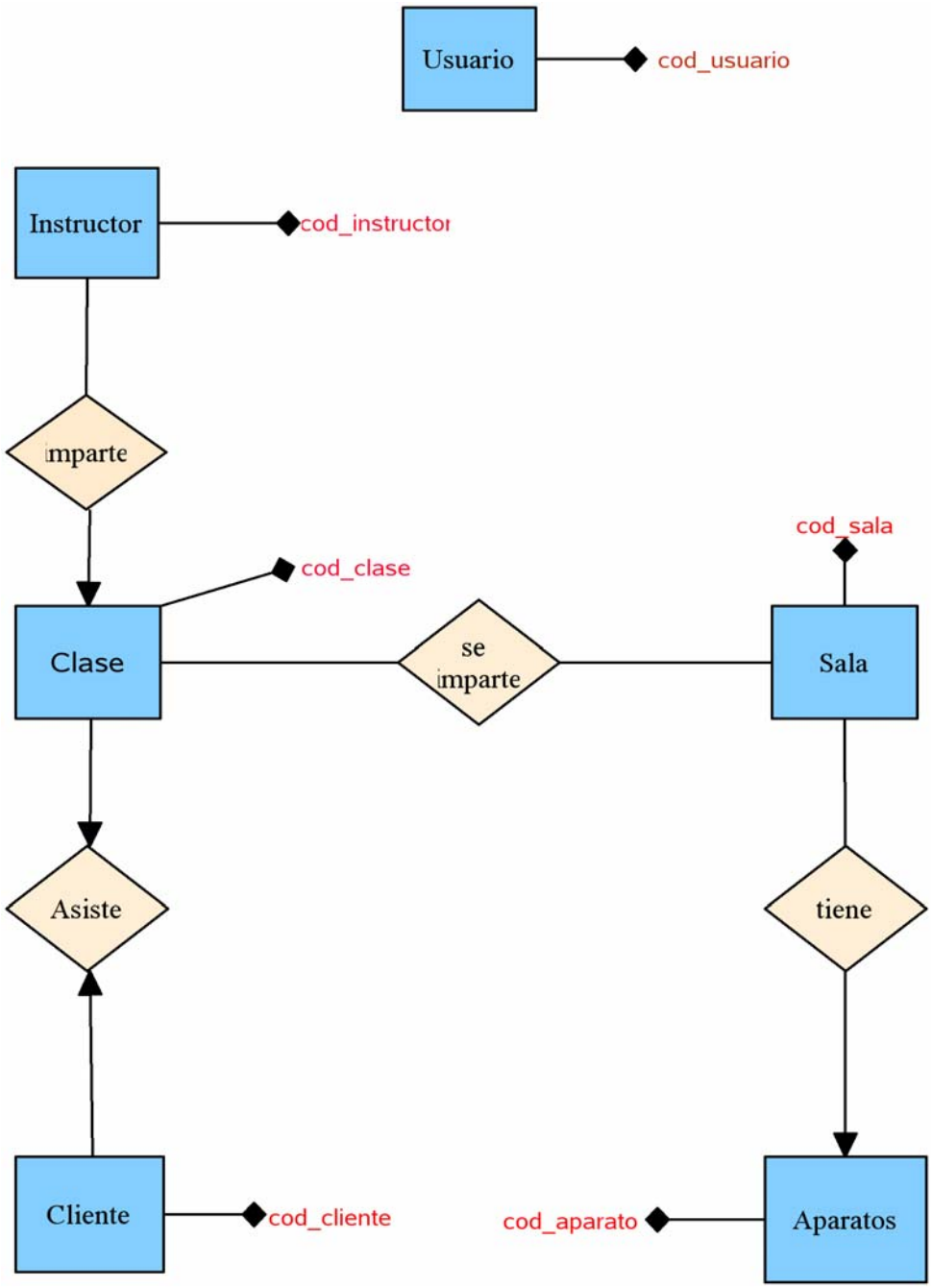

Fig. 15 Diagrama de flujo de datos “Acción Club”

Modelo Relacional de la aplicación "Registro y Control de Clientes del gimnasio Acción Club"

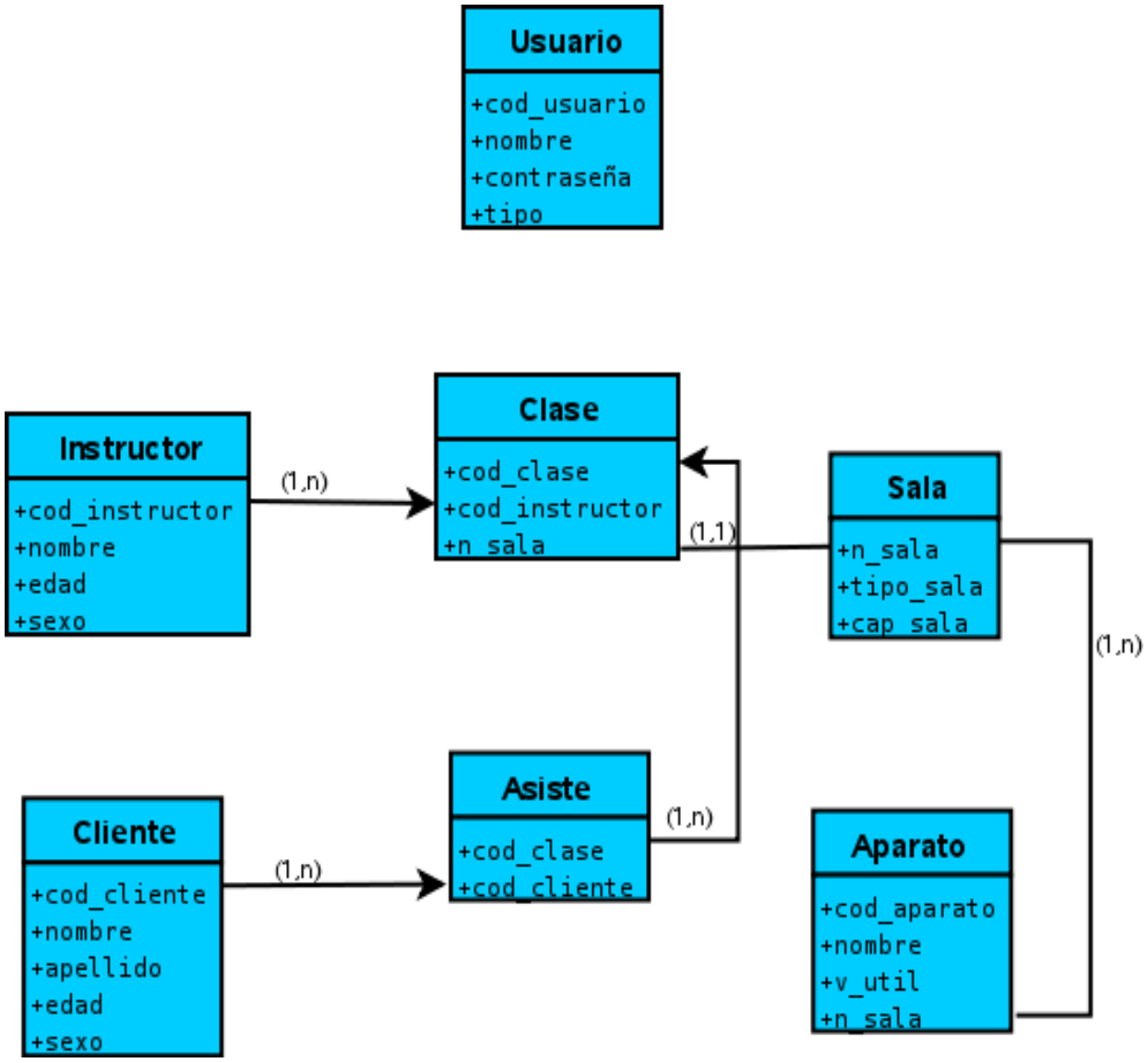

Fig. 16 Modelo Relacional "Acción Club"

Diseño Arquitectónico de la aplicación “Registro y Control de Clientes del gimnasio Acción Club”

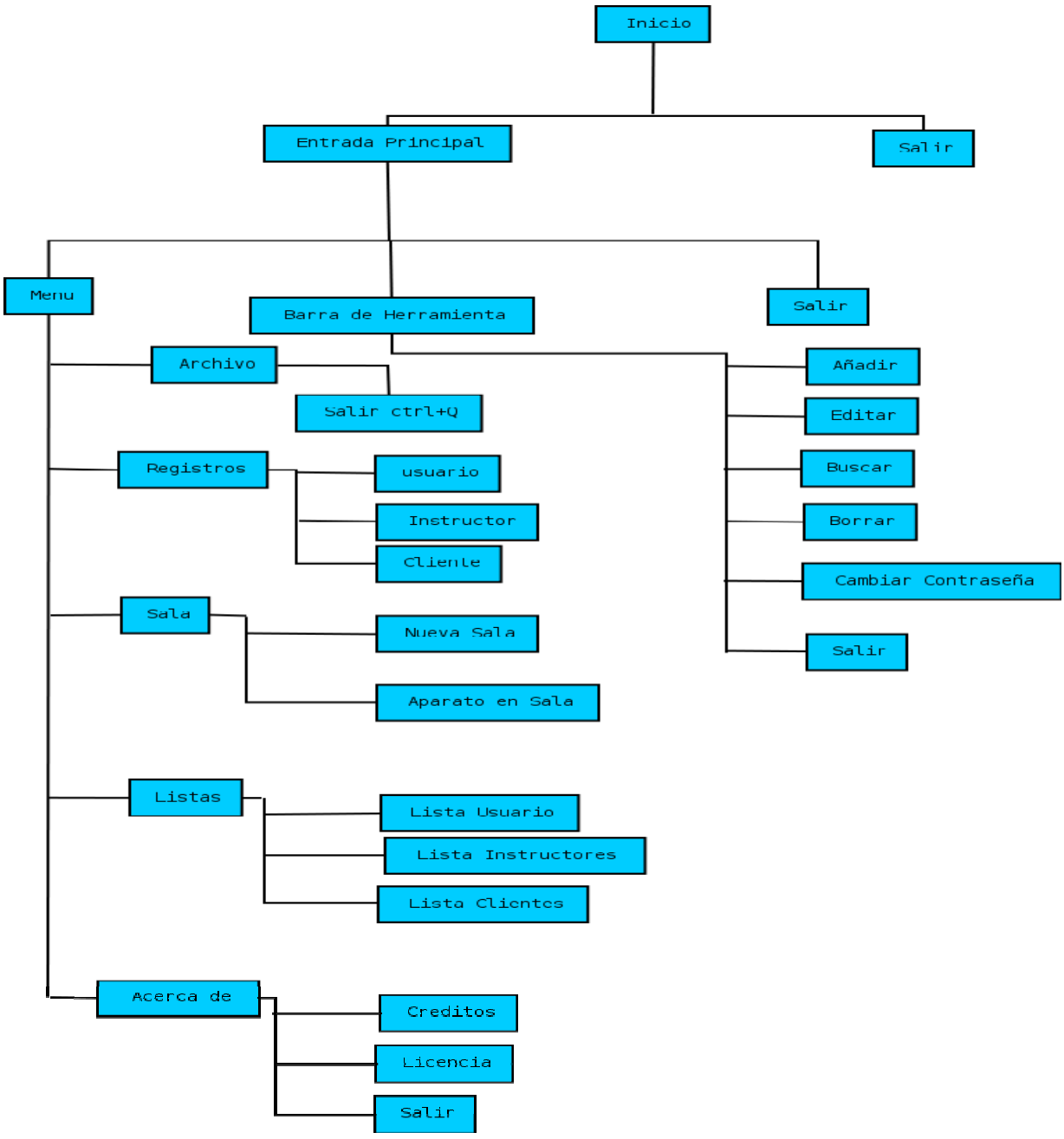

Fig.17 Diseño Arquitectónico “Acción Club”

Diagrama de Clase de la aplicación “Registro y control del gimnasio Acción Club”

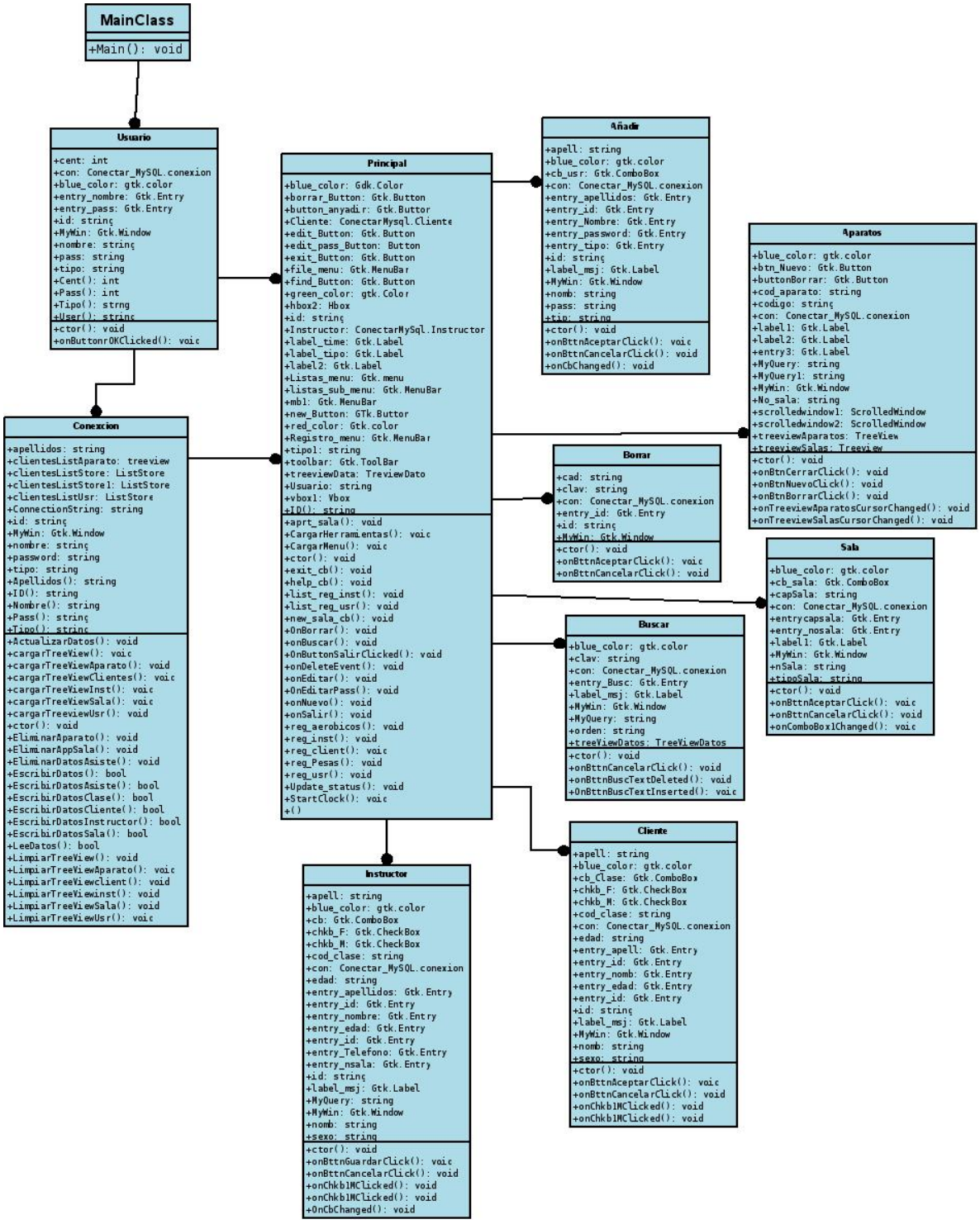

Fig. 18 Diagrama de Clase “Acción Club”

XI. Diseño de Interfaz

Este formulario es para la verificación de usuario en el cual se especificará el nombre de usuario y contraseña.

Fig. 19 Entrada de Usuario

Luego de comprobar el usuario es correcto nos llevará al formulario principal de la aplicación en donde encontraremos un menú y distintos íconos que nos mostrarán las funciones que se realizan.

Fig. 20 Formulario Principal

Con la **barra de herramientas** del principal podemos acceder de forma directa a los distintos formularios dentro de ellos encontramos un formulario que nos ayuda a cambiar nuestra contraseña.

Fig. 21 Barra de Herramientas

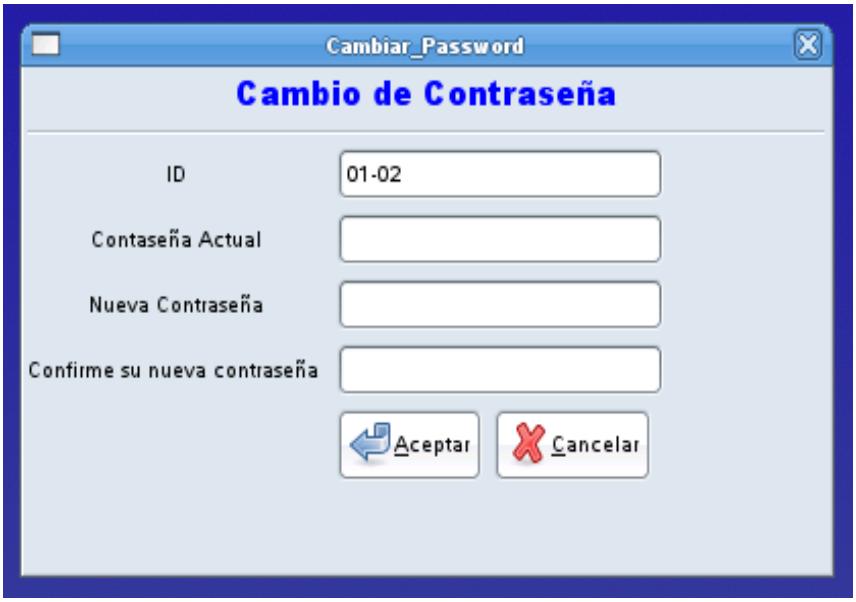

Fig. 22 Formulario Cambiar_Password

Con el **registro de usuario** podremos añadir, buscar, editar o borrar un usuario.

Fig. 23 Formulario Usuarios

Fig. 24 Formulario Añadir Usuario

Buscar Usuario este formulario realiza una búsqueda secuencial hasta encontrar al usuario.

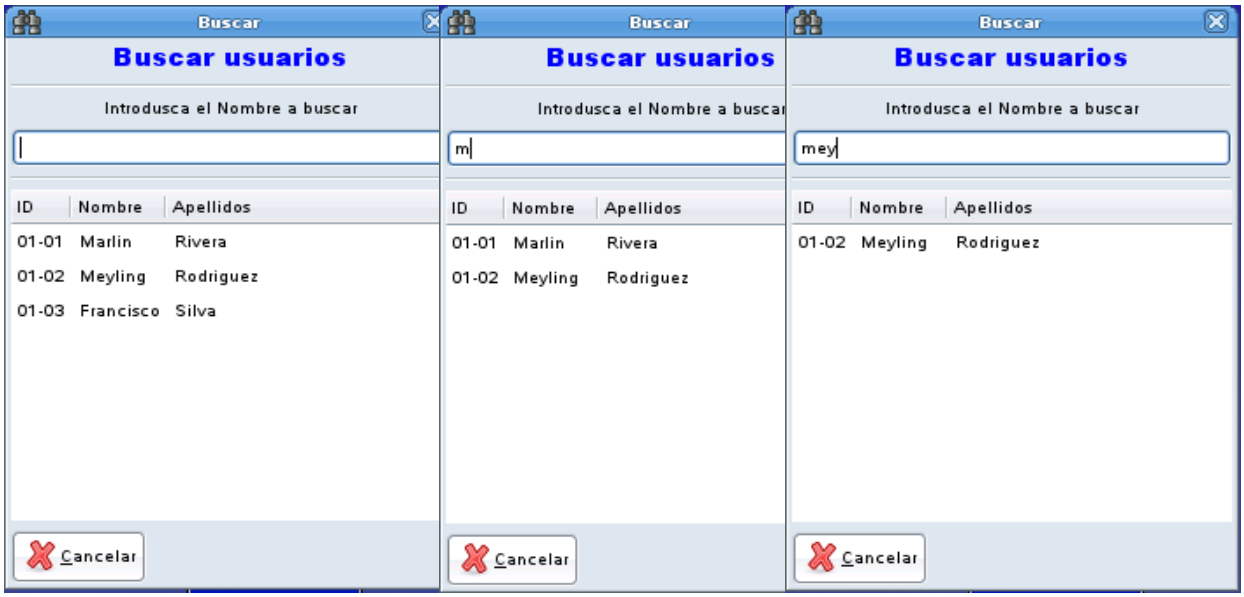

Fig. 25 Formulario Buscar

Con el formulario **edición** podemos editar cualquier registro existente en la base de datos de los usuarios, los clientes y los instructores.

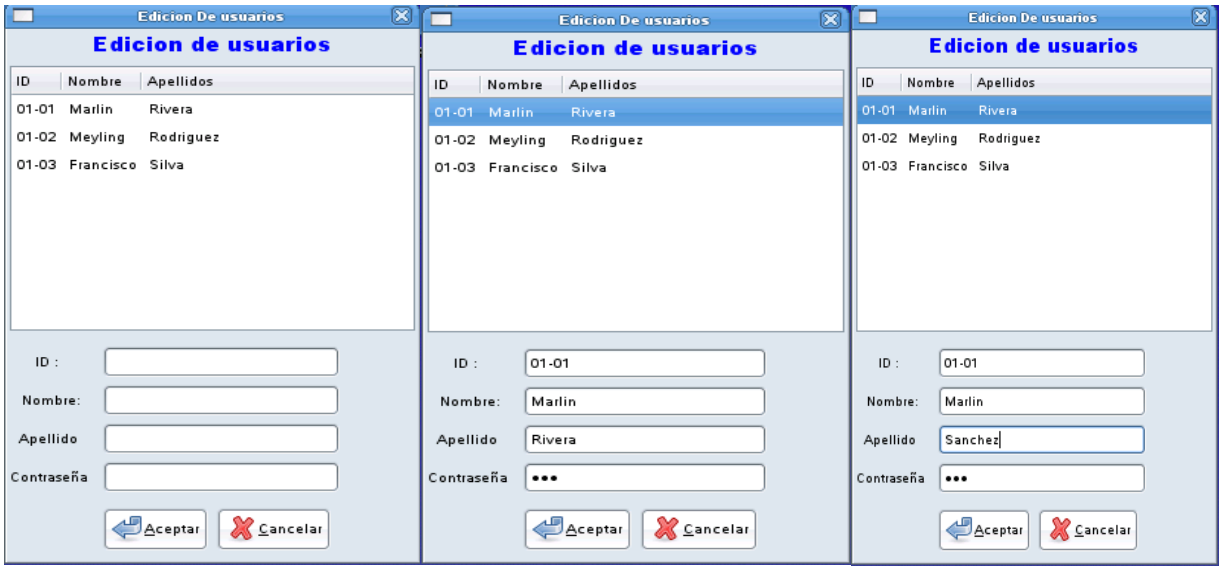

Fig. 26 Formulario Edición

En el formulario **Borrar** podemos borrar cualquier registro que se encuentre en la base de datos con la restricción de que solo se podrá borrar a través del id del registro.

Fig. 27 Formulario Borrar

En el formulario **instructor** encontramos las opciones añadir un nuevo instructor indicando su datos personales (como por ejemplo: nombre, edad, sexo, etc.), borrar, buscar, editar, y cancela, al igual que en el formulario de usuario.

Fig. 28 Formulario Instructor

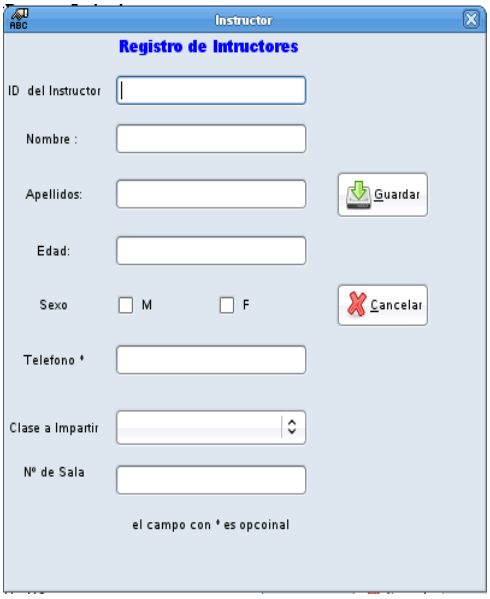

Fig. 29 Formulario Registro de Instructor

El formulario de **Clientes**, observamos los clientes actuales del gimnasio y además insertamos mas clientes si así se desea, buscamos un cliente determinado y borramos un cliente a través de su ID.

Fig. 30 Formulario Registro de Clientes

Fig. 31 Formulario de nuevo cliente

Con el formulario **sala** capturamos una información más detallada del local como cuántas salas existen, cuáles son de pesas y cuáles de aeróbicos y los aparatos existentes en cada sala.

Fig. 32 Formulario Sala

Con el formulario de **registro** de aparatos en sala verificamos en qué sala están los aparatos ya sean aparatos para aeróbicos o aparatos propios para pesas, así como su vida útil, además podemos agregar un nuevo aparato o borrar un aya existente.

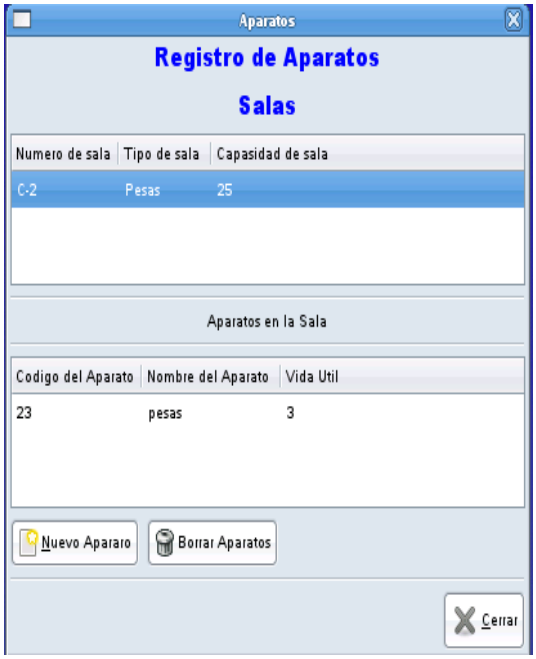

Fig. 33 Formulario Aparatos

Fig. 34 Formulario Nuevo _ aparato

El formulario de **Lista** contiene las listas de los usuarios y los instructores que existen en el gimnasio, además a que clases asisten e imparten.

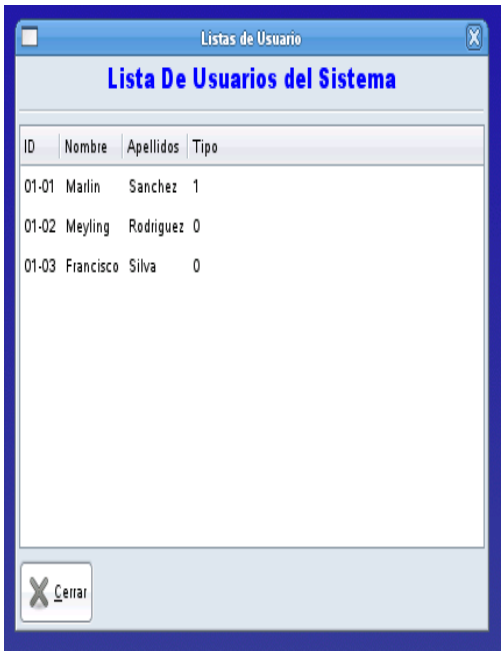

Fig. 35 Formulario Lista de usuario

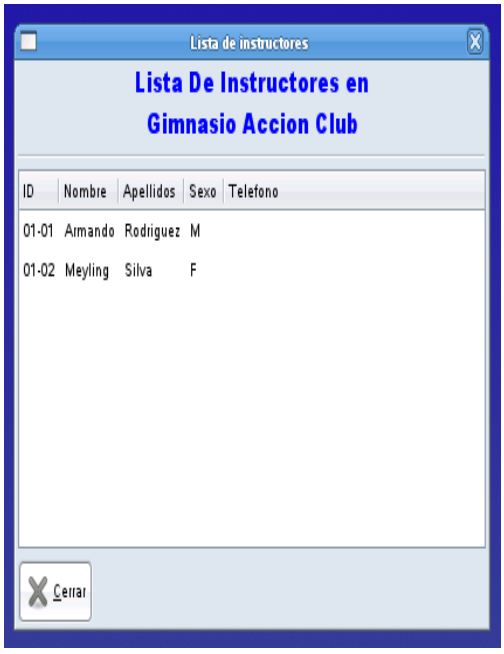

Fig. 36 Formulario Lista de Instructores

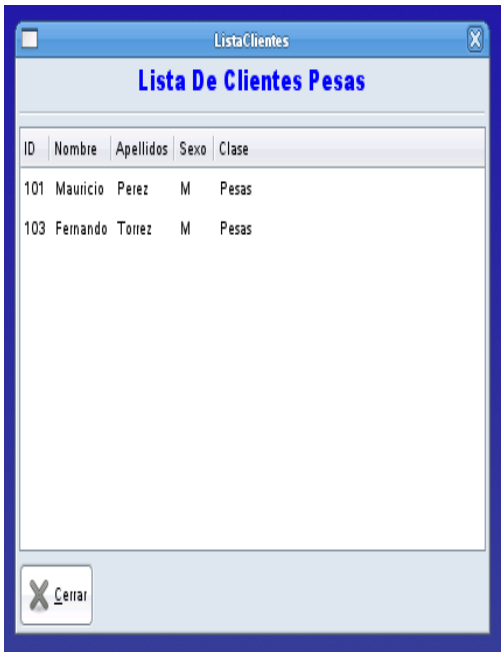

Fig. 37 Reporte de Clientes que asisten a clases de Pesas

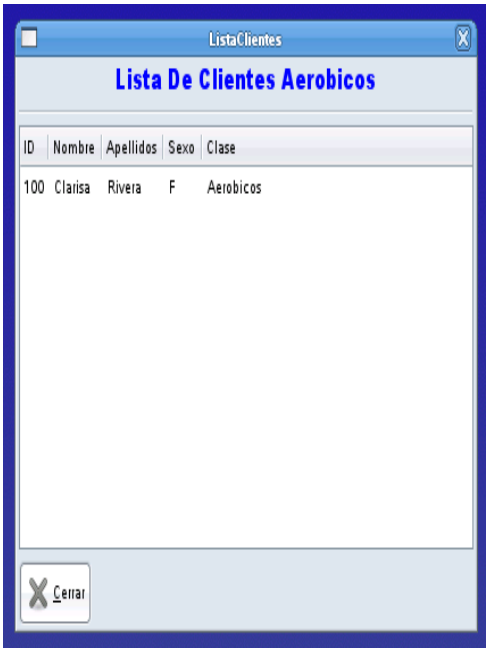

Fig. 38 Reporte de Clientes que asisten a clases de Aeróbicos

El formulario **Acerca de** contiene una breve información de la versión de Monodevelop, los autores y la licencia GPL.

Fig. 39 Formulario Acerca de

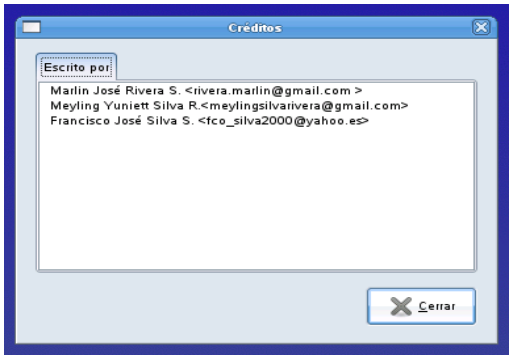

Fig. 40 Formulario de Créditos

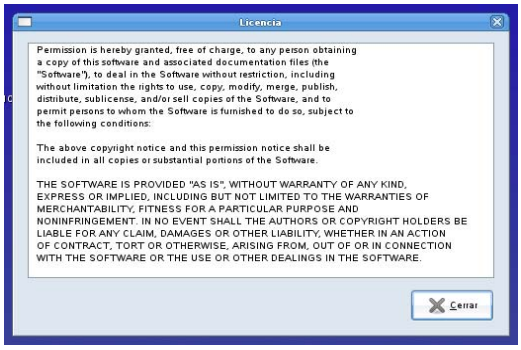

Fig. 41 Formulario de Licencia

XII. Codificación

A continuación mostramos formularios que se diseñaron para realizar la aplicación “Registro y Control de Clientes del gimnasio Acción Club”. Empezamos con el archivo conexión que nos conectan con nuestra base de dato.

```
using System;
using Gtk;
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
        public class Conexion
        {
        //cadena de conexión a la base de datos
        string connectionString =
        "Server=localhost;" +
        "Database=bd_gimnasio;" +
        "User ID=root;" +
        "Password=mya4ever;" +
        "Pooling=false";
        private string id;
        private string nombre;
        private string apellidos;
        private string password;
        private string tipo;
```

```
protected Gtk.Window MyWin;
//private string usuario;
//declaración de ListStore para llenar datos de los treeview
public Gtk.ListStore clientesListStore ;
public Gtk.ListStore clientesListStore1 ;
public Gtk.ListStore clientesListAparato ;
public Gtk.ListStore clientesListUsr ;
//declaracion de propiedades
        public string Nombre
        {
                get { return nombre; }
                set
                {
                        nombre = value;
                }
        }
public string Apellidos
        {
                get { return apellidos; }
                set
                {
                        apellidos = value;
                }
        }
```

```
public string Tipo {
        get { return tipo; }
        set { tipo= value; }
        }
public string ID
{
        get { return id; }
        set
        {
                id = value;
        }
    }
        public string Pass
        {
                get { return password; }
                set
                {
                        password = value;
                }
        }
//constructor
        public Conexion()
        {
        }
```

```
//Entrada de usuario
public bool LeerDatos(string nomb,string contr  )
{
        // Query para extraer la información de la tabla
        string myQuery = "SELECT * FROM usuarios;";
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        // Crea el objeto de ejecución del query
        MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
        // Ejecuta el query
        MySqlDataReader myReader = myCommand.ExecuteReader();
        // Lee cada uno de los registros devueltos
        while (myReader.Read())
        {
        if ((nomb == (""+myReader["nombre"])) && (contr == (""+myReader["password"])))  {
                //Console.WriteLine("Tipo: " + myReader["id"].ToString());
                Nombre= myReader["nombre"].ToString();
                Pass= myReader["password"].ToString();
                Tipo= myReader["tipo"].ToString();
                ID =myReader["id_usuario"].ToString();
```

```
                myReader.Close();
                return true;
            }
        }
        // Se cierra el reader
        myReader.Close();
        // Se cierra la conexión con la DB
        myConnection.Close();
        return false;
    }
    public bool EscribirDatos(string id ,string nomb,string apell,string pass,string tip)
     {
        //consulta MySQL para insertar datos en la tabla usuario
         string consulta _ escribir ="INSERT INTO usuarios
(id_usuario,nombre,apellidos,password,tipo) VALUES ('" + id + "','" + nomb  + "','"+
apell +"','"+ pass +"','"+ tip +"')";
        MySqlConnection myConnection = new
MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        try {
            MySqlCommand myCommand = new
MySqlCommand(consulta_escribir, myConnection);
            myCommand.ExecuteNonQuery();
            myConnection.Close();
```

```
                return true;
                }
        catch {
                MessageDialog msgBox = new
MessageDialogMyWin,DialogFlags.Modal,Mes
        sageType.Error,ButtonsType.Ok,"El Nombre del usuario ya existe ");
        ResponseType miResultado = (ResponseType)msgBox.Run ();
        msgBox.Destroy();
        if (miResultado == ResponseType.Ok){ }
        else { }
        return false;
        }
 }
public void EliminarDatos(string id,string cad)
 {
        //elimina datos del la tabla usuarios
        string consulta_escribir = cad;
        //"DELETE FROM usuarios" +" WHERE (id_usuario = '" + id + "');";
        MySqlConnection myConnection = new
MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        try {
                MySqlCommand myCommand = new
MySqlCommand(consulta_escribir, myConnection);
```

```
                myCommand.ExecuteNonQuery();
                myConnection.Close();
            }
        catch {
        // si hay alguna excepción se envía un mensaje
          MessageDialog msgBox = new MessageDialog(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Usuario no se puede Eliminar ");
        ResponseType miResultado = (ResponseType)msgBox.Run ();
        msgBox.Destroy();
        if (miResultado == ResponseType.Ok){ }
        else {}

        }
 }
public void ActualizarDatos(string myQuery)
{
        //edita los datos de la tabla usuarios
        string consulta_escribir =myQuery; //"UPDATE  usuarios SET nombre =  + '"+ Nomb  +"' , apellidos =   + '"+ apell + "' " + " WHERE(id_usuario =  '" + id +  "');";
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        //Abre la conexión
        myConnection.Open();
        try {
```

```
                MySqlCommand myCommand = new
MySqlCommand(consulta_escribir, myConnection);
                myCommand.ExecuteNonQuery();
                myConnection.Close();
            }
        catch{
                //si hay algún error envía un mensaje
                MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"No se pudo
Actualizar los Datos ");
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){}
                else {}
                }
}
//*****************Tabla Instructor*********************************///
public bool EscribirDatosInstructor(string id ,string nomb,string apell,string
edad,string sexo)
{
        //escribe los datos en la tabla instructor
        string consulta_escribir = "INSERT INTO instructor
(id_instructor,nombre,apellidos,edad,sexo) VALUES ('" + id + "','" + nomb + "','"+
apell +"','"+ edad +"','"+ sexo +"')";
        MySqlConnection myConnection = new
MySqlConnection(connectionString);
```

```
        // Abre la conexión
        myConnection.Open();
        try {
        // Se crea el comando mySql con la consulta y la cadena de conexión
                MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);
                myCommand.ExecuteNonQuery();
                myConnection.Close();
                return true;
            }
        catch{
        //si ocurre un error se manda un mensaje de error
                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Nombre del usuario ya existe ");
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){ }
                else {}
                return false;
                }
}
//*****************************Fin  de Instructor*********************************//
```

```
//*****************************Datos de Cliente*******************************//

public bool EscribirDatosCliente(string id ,string nomb,string apell,string edad,string sexo)

{

        //se crea la consulta mysql a escribir

        string consulta_escribir ="INSERT INTO clientes (id_cliente,nombre,apellidos,edad,sexo) VALUES ('" + id + "','" + nomb  + "','"+ apell +"','"+ edad +"','"+ sexo +"')";

        // se establece la conexión

        MySqlConnection myConnection = new MySqlConnection(connectionString);

        // Abre la conexión

        myConnection.Open();

        try {

                MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);

                myCommand.ExecuteNonQuery();

                myConnection.Close();

                return true;

            }

        catch {

                //si hay error envía un mensaje

                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Nombre del usuario ya existe ");

                ResponseType miResultado = (ResponseType)msgBox.Run ();
```

```
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){ }
                return false;
                }
}
//**************************Fin de Cliente*********************************//
//**************************Inicio De Sala*********************************//
public bool EscribirDatosSala(string nsala,string tipoSala,int capSala)
{
        //se crea la consulta mysql a escribir
        string consulta_escribir = "INSERT INTO sala (no_sala,tipo_sala,cap_sala) VALUES ('" + nsala + "','" + tipoSala  + "','"+ capSala + "');" ;
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        try {
                MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);
                myCommand.ExecuteNonQuery();
                myConnection.Close();
                return true;
              }
        catch(Exception e) {
        //si hay error se envía un mensaje
```

```
            MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"sala no insertada"+e.Message);
            ResponseType miResultado = (ResponseType)msgBox.Run ();
            msgBox.Destroy();
            if (miResultado == ResponseType.Ok){ }
            else { }
            return false; }
}
//*****************************Fin  de Sala*********************************//
//*****************************ASISTE*********************************//
public bool EscribirDatosAsiste(string id_cliente ,string cod_clase)
{
    //se crea la consulta mysql a escribir
    string consulta_escribir = " INSERT INTO asiste (id_cliente,cod_clase) VALUES ('" + id_cliente + "','" + cod_clase + "');" ;
    //se crea la  conexión
    MySqlConnection myConnection = new MySqlConnection(connectionString);
    // Abre la conexión
    myConnection.Open();
    try {
        MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);
        myCommand.ExecuteNonQuery();
```

```
                myConnection.Close();
                return true;
            }
        catch {
        //si hay error envía un mensaje
                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Nombre del usuario ya existe ");
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){ }
                else { }
                return false;
                }
}
//*****************************Fin  de Asiste*********************************//
//*****************************TreeViewData*********************************//
public void cargarTreeView(ref Gtk.TreeView treeviewDatos, string myQuery,string clav )
{
        //se crea el liststore con los parámetros que contendrá el treeview
        clientesListStore = new Gtk.ListStore (typeof(string), typeof(string), typeof(string), typeof(string), typeof(string));
        //se crea  el modelo para el treeview
        treeviewDatos.Model = clientesListStore;
```

```
// Se crea el objeto de conexión
MySqlConnection myConnection = new MySqlConnection(connectionString);
try {
    // Abre la conexión
    myConnection.Open();
    // Crea el objeto de ejecución del query
    MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
    // Ejecuta el query
    MySqlDataReader myReader = myCommand.ExecuteReader();
    // Lee cada uno de los registros devueltos
    while (myReader.Read())
    {
        // Se agrega una linea al treeview
        if(clav =="id_usuario"){
        clientesListStore.AppendValues(myReader[clav].ToString(), myReader["nombre"].ToString(),myReader["apellidos"].ToString(),myReader["password"].ToString()); }
    else {
        clientesListStore.AppendValues(myReader[clav].ToString(), myReader["nombre"].ToString(),myReader["apellidos"].ToString());
        }
    }
```

```
// Se cierra el reader
                myReader.Close();
               // Se cierra la conexión con la DB
                myConnection.Close();
          }
          catch(Exception myError) {
               // En caso de algún error lo veremos en la consola de depuración
               Console.WriteLine(myError.ToString());
               }
}
public void LimpiarTeeView() { clientesListStore.Clear(); }
/**********************ESCRIBIR DATOS CLASE ***************************/
public bool EscribirDatosClase(string cod_clase ,string id_inst, string n_sala)
{
     //se crea la consulta mysql a escribir
     string consulta_escribir ="INSERT INTO clase (cod_clase,id_instructor,no_sala) VALUES ('" + cod_clase + "','" + id_inst  + "','"+ n_sala +"')";
     //se crea el objeto de la conexión
     MySqlConnection myConnection = new MySqlConnection(connectionString);
     // Abre la conexión
     myConnection.Open();
     try {
          MySqlCommand myCommand = new
```

```
MySqlCommand(consulta_escribir, myConnection);
                myCommand.ExecuteNonQuery();
                myConnection.Close();
                return true;
            }
        catch {
        //si hay error se envía un mensaje
                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"La Clase no se Puede Crear");
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){ }
                else { }
                return false;
                }
}
/*******************************FIN DE CLASE**************************/
/*************ESCRIBIR DATOS APARATOS ***************************/
public bool EscribirDatosaAparatos(string cod_aparato ,string nombre_aparato,int v_util, string n_sala)
{
        //se crea la consulta mysql a escribir
        string consulta_escribir ="INSERT INTO aparatos (cod_aparato,nombre_aparato,v_util,no_sala) VALUES ('" + cod_aparato + "','" +
```

```
nombre_aparato  + "'," + v_util + "','"+ n_sala +"')";
        //se crea el objeto de la conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        try {
            //se crea el comando de mysql a ejecutar
            MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);
        //se ejecuta el query
            myCommand.ExecuteNonQuery();
            myConnection.Close();
            return true;
          }
        catch(Exception e) {
        //si ocurre un error se envía un mensaje
            MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El aparato no puede se Añadido" + e.Message);
            ResponseType miResultado = (ResponseType)msgBox.Run ();
            msgBox.Destroy();
            if (miResultado == ResponseType.Ok){ }
            else { }
            return false;
```

```
                }
}
/*****************************FIN DE APARATOS *************************/
/*****************************ELIMINAR APARATO *************************/
public bool EliminarAparato(string id)
{
        //se crea la consulta mysql a escribir
         string consulta_escribir ="DELETE FROM aparatos WHERE (cod_aparato = '" + id + "');";
        //s crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        // Abre la conexión
        myConnection.Open();
        try {
                MySqlCommand myCommand = new MySqlCommand(consulta_escribir, myConnection);
                myCommand.ExecuteNonQuery();
                myConnection.Close();
                return true;
           }
        catch(Exception e) {
        //si ocurre un error se envía un mensaje
                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El aparato no se
```

```
puede Eliminar "+ e.Message);
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){ return false; }
                return false;
                }
}
/*********************************FIN ELIMINAR APARATOS*********************/
/*****************************TREEVIEW DE SALAS*****************************/
public void cargarTreeViewSala(ref Gtk.TreeView treeviewDatos, string myQuery )
{
        //se crea el liststore para el treeview
        clientesListStore1 = new Gtk.ListStore (typeof(string), typeof(string), typeof(string));       treeviewDatos.Model = clientesListStore1;
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        try {
                // Abre la conexión
                myConnection.Open();
                // Crea el objeto de ejecución del query
                MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
                // Ejecuta el query
                MySqlDataReader myReader = myCommand.ExecuteReader();
```

```
                // Lee cada uno de los registros devueltos
                while (myReader.Read())
                {
                        // Se agrega una línea al treeview
            clientesListStore1.AppendValues(myReader["no_sala"].ToString(),
            myReader["tipo_sala"].ToString(),myReader["cap_sala"].ToString());
                }
                // Se cierra el reader
                myReader.Close();
                // Se cierra la conexión con la DB
                 myConnection.Close();
              }
            catch(Exception myError) {
                // En caso de algún error lo veremos en la consola de depuración
                Console.WriteLine(myError.ToString());
                }
}
public void LimpiarTeeViewSala() { clientesListStore1.Clear(); }
/*********************FIN DE TREEVIEW  SALA*******************/
/*********************TREEVIEW DE APARATOS*****************/
public void cargarTreeViewAparato(ref Gtk.TreeView treeviewDatos, string
myQuery )
{
        //Gtk.TreeView treeviewDatos=new Gtk.TreeView();
      clientesListAparato = new Gtk.ListStore (typeof(string), typeof(string),
```

```
typeof(string));        treeviewDatos.Model = clientesListAparato;
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        try {
                // Abre la conexión
                myConnection.Open();
                // Crea el objeto de ejecución del query
                MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
                // Ejecuta el query
                MySqlDataReader myReader = myCommand.ExecuteReader();
                // Lee cada uno de los registros devueltos
                while (myReader.Read())
                {
                // Se agrega una línea al treeview
 clientesListAparato.AppendValues(myReader["cod_aparato"].ToString(), myReader["nombre_aparato"].ToString(),myReader["v_util"].ToString());
                //Console.WriteLine("ID: " + myReader["id_usuario"].ToString());
                }
                // Se cierra el reader
                 myReader.Close();
                // Se cierra la conexión con la DB
                 myConnection.Close();
            }
```

```
        catch(Exception myError) {
                // En caso de algún error lo veremos en la consola de depuración
                Console.WriteLine(myError.ToString());
                }
}
public void LimpiarTeeViewAparato() { clientesListStore1.Clear(); }
/*********************FIN DE TREEVIEW  APARATOS**********************/
/*************************TREEVIEW DE USR*******************************/
public void cargarTreeViewUsr(ref Gtk.TreeView treeviewDatos, string myQuery )
{
        //Gtk.TreeView treeviewDatos=new Gtk.TreeView();
       this.clientesListUsr= new Gtk.ListStore (typeof(string), typeof(string), typeof(string), typeof(string), typeof(string),typeof(string));
        treeviewDatos.Model = clientesListUsr;
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        try {
                // Abre la conexión
                myConnection.Open();
                // Crea el objeto de ejecución del query
                MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
                // Ejecuta el query
                MySqlDataReader myReader = myCommand.ExecuteReader();
```

```
                // Lee cada uno de los registros devueltos
                while (myReader.Read())
                {
                        // Se agrega una línea al treeview
clientesListUsr.AppendValues(myReader["id_usuario"].ToString(),
myReader["nombre"].ToString(),myReader["apellidos"].ToString(),myReader["pass
word"].ToString(),myReader["tipo"].ToString());
        }
                        // Se cierra el reader
                         myReader.Close();
                        // Se cierra la conexión con la DB
                         myConnection.Close();
           }
        catch(Exception myError) {
                // En caso de algún error lo veremos en la consola de depuración
                Console.WriteLine(myError.ToString());
                }
}
public void LimpiarTeeViewUsr() {         clientesListUsr.Clear(); }
/******************FIN DE TREEVIEW  USR*******************/
/******************TREEVIEW LIST INSTRUCTOR************/
public void cargarTreeViewInst(ref Gtk.TreeView treeviewDatos, string myQuery )
{
        //Gtk.TreeView treeviewDatos=new Gtk.TreeView();
      this.clientesListUsr=  new  Gtk.ListStore  (typeof(string),  typeof(string),
```

```
typeof(string), typeof(string),typeof(string),typeof(string));
        treeviewDatos.Model = clientesListUsr;
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new MySqlConnection(connectionString);
        try {
            // Abre la conexión
            myConnection.Open();
            // Crea el objeto de ejecución del query
            MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
            // Ejecuta el query
            MySqlDataReader myReader = myCommand.ExecuteReader();
            // Lee cada uno de los registros devueltos
            while (myReader.Read())
            {
                // Se agrega una linea al treeview
            clientesListUsr.AppendValues(myReader["id_instructor"].ToString(), myReader["nombre"].ToString(),myReader["apellidos"].ToString(),myReader["edad"].ToString(),myReader["sexo"].ToString(),myReader["tlf_instructor"].ToString());
                //Console.WriteLine("ID: " + myReader["id_usuario"].ToString());
            }
// Se cierra el reader
             myReader.Close();
```

```
                // Se cierra la conexión con la DB
                 myConnection.Close();
            }
        catch(Exception myError) {
                // En caso de algún error lo veremos en la consola de depuración
                Console.WriteLine(myError.ToString());
                }
}
public void LimpiarTeeViewinst() {        clientesListUsr.Clear(); }
/**********************fin de treeviewinstructor*****************************/
/**********************Treeview list instructor*****************************/
public void cargarTreeViewCliente(ref Gtk.TreeView treeviewDatos, string
myQuery )
{
        //Gtk.TreeView treeviewDatos=new Gtk.TreeView();
        this.clientesListUsr= new Gtk.ListStore (typeof(string), typeof(string),
typeof(string), typeof(string),typeof(string),typeof(string));
        treeviewDatos.Model = clientesListUsr;
        // Se crea el objeto de conexión
        MySqlConnection myConnection = new
MySqlConnection(connectionString);
        try {
                // Abre la conexión
                myConnection.Open();
                // Crea el objeto de ejecución del query
```

```
MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
// Ejecuta el query
                MySqlDataReader myReader = myCommand.ExecuteReader();
                // Lee cada uno de los registros devueltos
                while (myReader.Read())  {        // Se agrega una línea al treeview
                clientesListUsr.AppendValues(myReader["id_cliente"].ToString(),
myReader["nombre"].ToString(),myReader["apellidos"].ToString(),myReader["edad"
].ToString(),myReader["sexo"].ToString(),myReader["cod_clase"].ToString());
                        //Console.WriteLine("ID:                  "                  +
myReader["id_usuario"].ToString());
                }
                // Se cierra el reader
                 myReader.Close();
                // Se cierra la conexión con la DB
                 myConnection.Close();
            }
        catch(Exception myError) {
                // En caso de algún error lo veremos en la consola de depuración
                Console.WriteLine(myError.ToString());
                }
}
public void LimpiarTeeViewclient() { clientesListUsr.Clear(); }
/******************Fin de treeviewinstructor ********************/
        }//fin de class
}// fin de Conexión .cs
```

Formulario Usuario.cs

```
using System;

namespace ConectarMySQL
{       public class Usuario : Gtk.Dialog
        {
        //declaración de variables y propiedades
                protected Gtk.Entry entry_nombre;
                protected Gtk.Entry entry_pass;
                string pass;
                string nombre;
                string tipo;
                string id;
        public string ID {
                get { return id; }
                set { id= value; }
                }
        public string User {
                get { return nombre; }
                set { nombre= value; }
                }
        public string Tipo {
                get { return tipo; }
                set { tipo= value; }
                }
```

```
public string Pass {
        get { return pass; }
        set {pass=value; }
        }
//declaración de variable tipo conexión
public ConectarMySQL.Conexion Con =new  ConectarMySQL.Conexion();
public Usuario()
        {
                Stetic.Gui.Build(this, typeof(ConectarMySQL.Usuario));
        }
protected virtual void OnButtonOkClicked (object sender, System.EventArgs e)
        {
                nombre= entry_nombre.Text;
                pass= entry_pass.Text;
        //compara si los datos existen en la base de datos
        //los almacena en las propiedades
                if(Con.LeerDatos(nombre,pass))
                {
                        User=Con.Nombre;
                        Pass=Con.Pass;
                        Tipo=Con.Tipo;
                        ID= Con.ID;
                }
```

```
else {
                        User="";
                        Pass="";
                }
        }
        }
}// Fin del archivo usuario.cs
```

Formulario: Principal.cs

```
using System;
using System.IO;
using Gtk;
using Gdk;
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
        public class Principal :Gtk.Window
        {
                protected Gtk.VBox vbox1;
                //declaración de variables
                protected Gtk.Label label2;
                //colores para las etiquetas
                Gdk.Color green_color = new Gdk.Color (0 , 0xaa,0 );
```

```
Gdk.Color red_color = new Gdk.Color (0xff, 0, 0);
Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
//componentes de la interfaz principal
protected Gtk.Button button_anyadir;
protected Gtk.MenuBar mb1;
 protected Gtk.Label label_time;
protected Gtk.Label label1;
 protected Gtk.Toolbar toolbar;
protected Gtk.Label label_tipo;
public Menu file_menu = new Menu ();
public Menu Registro_menu = new Menu ();
public Menu Sala_menu = new Menu ();
public Menu Listas_menu = new Menu();
public Menu listas_sub_menu = new Menu ();
//Botones de Barra de herramientas
public ToolButton exit_Button = new ToolButton (Stock.Close);
public ToolButton new_Button = new ToolButton (Stock.New);
public ToolButton find_Button = new ToolButton (Stock.Find);
public ToolButton borrar_Button = new ToolButton (Stock.Delete);
public ToolButton edit_Button = new ToolButton (Stock.Edit);
public ToolButton edit_pass_Button = new ToolButton (Stock.SpellCheck);
 //String
public string tipo1;
public string id="";
```

```
public string ID
        {
            get { return id; }
            set { id = value; }
        }
    //constructor de la clase principal
    public Principal (string Id,string usr,string tipo): base ("")
    {
        Stetic.Gui.Build (this, typeof(ConectarMySQL.Principal));
         this.StartClock();
        label_time.ModifyFg(Gtk.StateType.Normal,blue_color );
        label_time.ModifyBg(Gtk.StateType.Normal,red_color );
        label1.ModifyFont(Pango.FontDescription.FromString("Adobe Utopia
14"));
        this.SetDefaultSize(800,800);
        tipo1=tipo;
        this.ID=Id;
        //cambiar el color de la fuente
        label1.ModifyFg(Gtk.StateType.Normal,green_color);
        this.label2.ModifyFg(Gtk.StateType.Normal,red_color );
        label2.Text="Nombre de Usuario : " + usr;
        this.label_tipo.ModifyFg(Gtk.StateType.Normal,red_color );
        label_tipo.Text= "Tipo de Usuario " + tipo;
        CargarMenu();
        // Función para cargar la barra de herramientas
```

```
            this.CagarHerramientas();
            if(tipo=="Usuario")
            {
                // Si es de tipo usuario deshabilitar botones
                this.edit_Button.Sensitive = false;
                this.borrar_Button.Sensitive= false;
                this.new_Button.Sensitive=false;
            }
        }
         public void CargarMenu()
        {
            AccelGroup grup = new AccelGroup ();
            AddAccelGroup(grup);
        /******************Barra de Menú*********************/
         /******************Menú Archivo*********************/
        //Orden del menú Salir
         ImageMenuItem quit_item = new ImageMenuItem(Stock.Quit, grup);
        quit_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.q,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
         quit_item.Activated += new EventHandler (exit_cb);
         file_menu.Append(quit_item);
         MenuItem file_item = new MenuItem("Archivo");
        file_item.Submenu = file_menu;
        mb1.Append (file_item);
         if(this.tipo1!="Usuario"){
```

```
/**********************MENU REGISTROS**************************/
 //Nuevo Usuario
 MenuItem usr_item = new MenuItem("_Usuario");
usr_item.Activated += new EventHandler (reg_usr);
usr_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.u,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
Registro_menu.Append (usr_item);
 // Instructor
MenuItem Inst_item = new MenuItem("Instructor");
Inst_item.Activated += new EventHandler (reg_inst);
Registro_menu.Append (Inst_item);
//Clientes
MenuItem client_item = new MenuItem("_Cliente");
client_item.Activated += new EventHandler (reg_client);
Registro_menu.Append (client_item);
MenuItem reg_item = new MenuItem("Registros");
reg_item.Submenu = Registro_menu;
mb1.Append (reg_item);
/********************** FIN MENU REGISTROS**************************/

/******************Menú Sala*********************/
//Orden del menú Sala
ImageMenuItem new_item = new ImageMenuItem(Stock.New, grup);
new_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.N,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
```

```
new_item.Activated += new EventHandler (new_sala_cb);
Sala_menu.Append(new_item);
 //orden de Aparatos en sala
MenuItem aprt_item = new MenuItem("Aparatos en Salas");
aprt_item.Activated += new EventHandler (aprt_sala);
aprt_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.A,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
Sala_menu.Append (aprt_item);
MenuItem sala_item = new MenuItem("Sala");
sala_item.Submenu = Sala_menu;
mb1.Append (sala_item);
}
/**********************MENU Listas**************************/
  //lista Usuario
MenuItem list_usr_item = new MenuItem("_Usuarios");
list_usr_item.Activated += new EventHandler (list_reg_usr);
list_usr_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.L,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
Listas_menu.Append (list_usr_item);
//Instructor
MenuItem list_Inst_item = new MenuItem("_Instructor");
list_Inst_item.Activated += new EventHandler (list_reg_inst);
Listas_menu.Append (list_Inst_item);
//Pesas
MenuItem list_pesas_item = new MenuItem("_Clientes en Pesas");
```

```
        list_pesas_item.Activated += new EventHandler (reg_Pesas);
        listas_sub_menu.Append (list_pesas_item);
        //Aeróbicos
        MenuItem list_aerobicos_item = new MenuItem("_Clientes en Aeróbicos");
        list_aerobicos_item.Activated += new EventHandler (reg_aerobicos);
        listas_sub_menu.Append (list_aerobicos_item);
        MenuItem list_item = new MenuItem("Asisten...");
        list_item.Submenu = listas_sub_menu;
        Listas_menu.Append(list_item);
        MenuItem list_item1 = new MenuItem("Listas");
        list_item1.Submenu = Listas_menu;
        mb1.Append (list_item1);
        /********************** FIN MENU Listas**************************/
        //Ayuda
        ImageMenuItem help_item = new ImageMenuItem(Stock.About,grup);
        help_item.AddAccelerator("activate", grup, new AccelKey(Gdk.Key.h,
Gdk.ModifierType.ControlMask, AccelFlags.Visible));
        help_item.Activated += new EventHandler (help_cb);
        mb1.Append(help_item);
/***************************FIN MENU SALA*******************************/
        mb1.ShowAll();
        }
                public void CagarHerramientas()
        {
```

```
/************************BARRA DE HERRAMIENTAS************************/
            exit_Button.IsImportant = true;
            exit_Button.Clicked +=OnSalir;
            toolbar.Insert (exit_Button, -1);
            new_Button.IsImportant = true;
            new_Button.Clicked +=OnNuevo;
            toolbar.Insert (new_Button, -1);
            // Botón editar
            edit_Button.IsImportant = true;
            edit_Button.Clicked +=OnEditar;
            toolbar.Insert (edit_Button, -1);
            ToolButton find_Button = new ToolButton (Stock.Find);
            find_Button.IsImportant = true;
            find_Button.Clicked +=OnBuscar;
            toolbar.Insert (find_Button, -1);
            borrar_Button.IsImportant = true;
            borrar_Button.Clicked +=OnBorrar;
            toolbar.Insert (borrar_Button, -1);
            edit_pass_Button.IsImportant = true;
            edit_pass_Button.Clicked +=OnEditarPass;
            toolbar.Insert (edit_pass_Button, -1);
            toolbar.ShowAll();
/************************FIN DE BARRA DE HERRAMIENTAS***************/
}
```

```
static void exit_cb (object o, EventArgs args)
{  Application.Quit (); }

/**********************HELP*************************/
static void help_cb (object o, EventArgs args)
 {
          new ConectarMySQL.Ayuda();
}
/*****INICIO DE MANEJADORES DE MENU REGISTROS*********/
static void reg_usr (object o, EventArgs args)
{
        ConectarMySQL.Editar Act =new ConectarMySQL.Editar("usuarios");
        Act.Show ();
 }
static void reg_inst (object o, EventArgs args)
{
ConectarMySQL.Editar Act =new ConectarMySQL.Editar("instructor");
        Act.Show ();
 }
static void reg_aerobicos (object o, EventArgs args)
{
        new ConectarMySQL.ListaClientes("Aerobicos");
}
```

```
static void reg_Pesas(object o, EventArgs args)
{
        new ConectarMySQL.ListaClientes("Pesas");
}
static void reg_client (object o, EventArgs args)
{
        ConectarMySQL.Editar Act =new ConectarMySQL.Editar("clientes");
        Act.Show ();
 }
/************ FIN DE REGISTROS**********************************/
/**************MANEJADORES DEL MENU LISTAS****************/
 static void list_reg_usr (object o, EventArgs args)
 {
        new ConectarMySQL.ListasUsr();
}
static void list_reg_inst (object o, EventArgs args)
{
         new ConectarMySQL.ListInst();
}
/****************************FIN DE LISTAS************************************/
 /***********INICIO DE MANEJADORES DE MENU SALA******************/
 static void new_sala_cb (object o, EventArgs args)
{
        ConectarMySQL.Sala sal = new ConectarMySQL.Sala();
        sal.Show();             }
```

```
        static void aprt_sala (object o, EventArgs args)
        {
                ConectarMySQL.Aparatos sal = new ConectarMySQL.Aparatos();
                sal.Show();
        }
/*************************FIN DE SALA************************************/
/*************************Manejadores Barra de Herramientas***************/
        static void OnBuscar (object o, EventArgs args)
        {
                new    ConectarMySQL.Buscar("usuarios");
        }
        static void OnBorrar (object o, EventArgs args)
        {
                new ConectarMySQL.Borrar("usuarios");
        }
        static void OnNuevo (object o, EventArgs args)
        {
                 new ConectarMySQL.Añadir();
        }
        static void OnEditar(object o, EventArgs args)
        {
                 new ConectarMySQL.Edicion("usuarios");
        }
```

```
        public void OnEditarPass(object o, EventArgs args)
        {
                string valor=this.ID;
                new ConectarMySQL.Cambiar_Password(valor);
         }
        static void OnSalir (object o, EventArgs args)
         {
                Application.Quit();
        }
/*************************Fin de Barra de Herramientas***********/
protected virtual void OnButtonSalirClicked (object sender, System.EventArgs e)
        {
                // Se cierra la Aplicación
                Application.Quit();
        }
        protected void OnDeleteEvent (object sender, DeleteEventArgs a)
        {
                Application.Quit ();
                a.RetVal = true;
        }
        //Reloj
         void StartClock ()
        { GLib.Timeout.Add(100, new GLib.TimeoutHandler(update_status)); }
```

```
        bool update_status ()
        {
                // Se establece la etiqueta con la fecha y Hora
                label_time.Text= "Fecha y Hora :" + DateTime.Now.ToString ();
                return true;
        }
  }
}//fin del principal
```

Main.cs

```
using System;
using Gtk;
namespace ConectarMySQL
        {
        class MainClass
        {
                public static void Main (string[] args)
                {
                        Application.Init (); // Se inicia la Aplicación
                        //Se declara la ventana principal
                        ConectarMySQL.Principal win;
                // se llama el formulario de usuario
                ConectarMySQL.Usuario usr=new ConectarMySQL.Usuario();
                        ResponseType MyResponse = (ResponseType)usr.Run();
```

```
        if (MyResponse==ResponseType.Ok)
                        {
                                if (usr.User!="" && usr.Pass!="" )
                                {
                        win= new ConectarMySQL.Principal(usr.ID,usr.User,usr.Tipo);
                                usr.Destroy();
                                win.Show();
                                Application.Run ();
                                }
                        }
                        else if(MyResponse==ResponseType.Cancel)
                        {
                                usr.Destroy();
                        }
                }//Main fn
        }//end class
}//end namespace
```

AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
// Information about this assembly is defined by the following
// attributes.
// // change them to the information which is associated with the assembly
// you compile.
```

```
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
// The assembly version has following format :
// Major.Minor.Build.Revision
// You can specify all values by your own or you can build default build and revision
// numbers with the '*' character (the default):
[assembly: AssemblyVersion("1.0.*")]
// The following attributes specify the key for the sign of your assembly. See the
// .NET Framework documentation for more information about signing.
// This is not required, if you don't want signing let these attributes like they're.
[assembly: AssemblyDelaySign(false)]
[assembly:AssemblyKeyFile("")]
```

Cambiar_Password.cs

```
using System;
using Gtk;
namespace ConectarMySQL
{
```

```
public class Cambiar_Password : Gtk.Window
{
        protected Gtk.Entry entry_old_pass;
        protected Gtk.Entry entry_new_pass;
        protected Gtk.Entry entry_confirm;
        public string myQuery;
        public string pass;
        public string id;
        protected Gtk.Window MyWin;
        ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
        protected Gtk.Entry entry_id;
         Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
         protected Gtk.Label label_msj;
        public Cambiar_Password(string Id) :
base(Gtk.WindowType.Toplevel)
        {
                Stetic.Gui.Build(this,
typeof(ConectarMySQL.Cambiar_Password));
                this.label_msj.ModifyFg(Gtk.StateType.Normal,blue_color );
          label_msj.ModifyFont(Pango.FontDescription.FromString("Arial Black
14"));
                id=Id;
                entry_id.Text=Id;
        }
```

```
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs
e)		{
			this.Destroy();
		}
		protected virtual void OnButtonAceptarClicked (object sender,
System.EventArgs e)
		{
		// Edita  datos por medio del ID
			pass=entry_new_pass.Text;
			myQuery=  "UPDATE  usuarios SET password= + "'+ pass + "'
" + " WHERE(id_usuario =  '" + id +  "');";
			if (entry_new_pass.Text == entry_confirm.Text)
			{
				con.ActualizarDatos(myQuery);
				MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"La contraseña ha
sido cambiada  \n En el Registro ");
				msgBox.Show();
				ResponseType miResultado =
(ResponseType)msgBox.Run ();
				msgBox.Destroy();
				if (miResultado == ResponseType.Ok)
				{
				 this.Destroy();
				}
			}
```

```
                    else{
                        //si hay error envía un mensaje
                        MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"Las Contraseñas
no coinsiden ");
                        msgBox.Show();
                        ResponseType miResultado =
(ResponseType)msgBox.Run ();
                        msgBox.Destroy();
                        if (miResultado == ResponseType.Ok)
                        {
                         //this.Destroy();
                            entry_old_pass.Text = "";
                            entry_new_pass.Text = "";
                            entry_confirm.Text  = "";
                        }
                    }
                }
            }
} //fin de Cambiar_Password.cs
```

Registro de Usuario: Añadir.cs

```
using System;
using Gtk;
```

```
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
public class Añadir : Gtk.Window
{
        protected Gtk.Entry entry_id;
        protected Gtk.Entry entry_apellidos;
        protected Gtk.Entry entry_Nombre;
        string id;
        string nomb;
        string apell;
        string pass;
        string tip;
         protected Gtk.Window MyWin;
        public ConectarMySQL.Conexion con= new ConectarMySQL.Conexion();
        protected Gtk.Entry entry_password;
        protected Gtk.Entry entry_tipo;
        protected Gtk.Label label_msj;
        Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
        protected Gtk.ComboBox cb_usr;
```

```
public Añadir() : base(Gtk.WindowType.Toplevel)
{
        Stetic.Gui.Build(this, typeof(ConectarMySQL.Añadir));
        label_msj.ModifyFont(Pango.FontDescription.FromString("Arial Black 14"));
        label_msj.ModifyFg(Gtk.StateType.Normal,blue_color );
}
protected virtual void OnBttnAceptarClicked (object sender, System.EventArgs e)
{
        if(entry_id.Text.ToString().Length == 0 ||entry_Nombre.Text.ToString().Length == 0 || entry_apellidos.Text.ToString().Length == 0||entry_password.Text.ToString().Length == 0)
        {
                MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Usuario no se puede Añadir \n Faltan Datos ");
                msgBox.Show();
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){
                entry_id.Text="";
                entry_Nombre.Text="";
                entry_apellidos.Text="";
```

```
                entry_password.Text="" ;
                //entry_tipo.Text="";
            }
            else {}
        }
        else{
            id=entry_id.Text;
            nomb=entry_Nombre.Text;
            apell=entry_apellidos.Text;
            pass=entry_password.Text ;
            //tip= entry_tipo.Text;
        if( con.EscribirDatos(id,nomb,apell,pass,tip))
        {
            MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"El Usuario ha sido
Añadido \n En el Registro ");
            msgBox.Show();
            ResponseType miResultado = (ResponseType)msgBox.Run ();
            msgBox.Destroy();
            if (miResultado == ResponseType.Ok)
            {
                entry_id.Text="";
                entry_Nombre.Text="";
                entry_apellidos.Text="";
                entry_password.Text="" ;
```

```
                    }
                else {}
        }
        else
        {
                entry_id.Text="";
                entry_Nombre.Text="";
                entry_apellidos.Text="";
                entry_password.Text="" ;
                //entry_tipo.Text="";
        }
   }
}
protected virtual void OnBttnCancelarClicked (object sender, System.EventArgs e)
{
        this.Destroy();
}
protected virtual void OnCbUsrChanged (object sender, System.EventArgs e)
        {
                TreeIter iter;
                if (cb_usr.GetActiveIter (out iter))
                tip=((string)cb_usr.Model.GetValue(iter,0));
        }
   } }//fin añadir.cs
```

Buscar.cs

```
using System;
using Gtk;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
public class Buscar : Gtk.Window
{
        //protected Gtk.Entry entrySeleccionado;
        protected Gtk.TreeView treeviewDatos;
        protected Gtk.Entry entry_Busc;
        protected Gtk.Expander expander1;
        ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
        public string clav;
        public string orden;
        public string myQuery;
      protected Gtk.ScrolledWindow scrolledwindow1;
      protected Gtk.Label label_msj;
      //Gdk.Color red_color = new Gdk.Color (0xff, 0, 0xff);
     Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
        public Buscar( string clave) :
        base(Gtk.WindowType.Toplevel)
        {
        orden=clave;
```

```
Stetic.Gui.Build(this, typeof(ConectarMySQL.Buscar));
// Se agregan las columnas al treeview
treeviewDatos.AppendColumn("ID", new Gtk.CellRendererText(), "text", 0);
treeviewDatos.AppendColumn("Nombre", new Gtk.CellRendererText(),
"text", 1);
treeviewDatos.AppendColumn("Apellidos", new Gtk.CellRendererText(),
"text", 2);
this.SetDefaultSize(400,400);
this.label_msj.Text="Buscar "+ clave;
label_msj.ModifyFont(Pango.FontDescription.FromString("Arial black 14"));
label_msj.ModifyFg(Gtk.StateType.Normal,blue_color);
scrolledwindow1.SetSizeRequest(250,200);//.SetSizeRequest(200,300);
//verifica en que tabla se va a buscar
if(clave == "usuarios")
{
        myQuery= "SELECT * FROM " + "usuarios";
        clav="id_usuario";
}
if(clave== "instructor")
{
        myQuery= "SELECT * FROM " + "instructor";
        clav="id_instructor";
}
```

```
if(clave == "clientes")
        {
                myQuery= "SELECT * FROM " + "clientes";
                clav="id_cliente";
        }
        //cargamos la base de datos en el TreeView
        con.cargarTreeView(ref treeviewDatos,myQuery,clav);
}//fin de buscar.cs
```

Edición.cs

```
using System;
using Gtk;
 // Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
        public class Edicion : Gtk.Window
        {
        protected Gtk.Entry entry_Id;
        protected Gtk.TreeView treeviewDatos;
        protected Gtk.Entry entry_Nombre;
        protected Gtk.Entry entry_apell;
        protected Gtk.Entry entry_pass;
```

```
protected Gtk.Label label4;
protected Gtk.Window MyWin;
string id;
string nomb;
string apell;
string pass;
ConectarMySQL.Conexion con = new ConectarMySQL.Conexion();
public string myQuery;
public string clave;
public string clav;
string Lista;
//Color del Texto
Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
protected Gtk.Label label_pass;
protected Gtk.ScrolledWindow scrolledwindow1;
public Edicion(string Clave) : base(Gtk.WindowType.Toplevel)
{
        Stetic.Gui.Build(this, typeof(ConectarMySQL.Edicion));
        this.Title= "Edición De "+Clave;
        clave=Clave;
        this.SetDefaultSize(400,400);
        scrolledwindow1.SetSizeRequest(250,200);
        treeviewDatos.AppendColumn("ID", new Gtk.CellRendererText(),
"text", 0);
        treeviewDatos.AppendColumn("Nombre", new
```

```
Gtk.CellRendererText(), "text", 1);
treeviewDatos.AppendColumn("Apellidos", new Gtk.CellRendererText(), "text", 2);
treeviewDatos.AppendColumn("Contaseña", new Gtk.CellRendererText(), "text",
3);
                treeviewDatos.Columns[3].Visible = false;
                //treeviewDatos.Columns[3].Visible = true;
                Lista= "SELECT * FROM " + Clave;
                label4.Text="Edición de "+ Clave;
                this.label4.ModifyFg(Gtk.StateType.Normal,blue_color );
                label4.ModifyFont(Pango.FontDescription.FromString("Arial Black
14"));
                if(clave == "usuarios")
                {
                        entry_pass.Show();
                        label_pass.Show();
                        clav="id_usuario";
                }
                if(clave== "instructor")
                {
                        clav="id_instructor";
                }
                if(clave == "clientes")
                {
                        clav="id_cliente";                      }
                con.cargarTreeView(ref treeviewDatos,Lista,clav);       }
```

```
protected virtual void OnButtonAceptarClicked (object sender, System.EventArgs
e)
        {
        if(entry_Id.Text.ToString().Length == 0
||entry_Nombre.Text.ToString().Length == 0 || entry_apell.Text.ToString().Length
== 0)  {
                MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"No se puede
Editar los Datos \n Faltan Datos ");
                msgBox.Show();
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){
                entry_Id.Text="";
                entry_Nombre.Text="";
                entry_apell.Text="";
                entry_pass.Text="" ;
                /entry_tipo.Text="";
                }
        }
        else{
                id=entry_Id.Text;
                nomb=entry_Nombre.Text;
                apell=entry_apell.Text;
                pass=entry_pass.Text ;
```

```
if(clave == "usuarios")
{
myQuery=  "UPDATE  usuarios SET nombre =  +  "'+ nomb +"' , apellidos = + "'+ apell + "' ,password= + "'+ pass + "'  " + " WHERE(id_usuario =  "' + id +  "');";
}

if(clave== "instructor")
{
myQuery=  "UPDATE  instructor SET nombre =  +  "'+ nomb +"' , apellidos =   + "'+ apell + "' " + " WHERE(id_instructor =  "' + id +  "');";
}
if(clave == "clientes")
{
myQuery=  "UPDATE  clientes SET nombre =  +  "'+ nomb  +"' , apellidos =   + "'+ apell + "' " + " WHERE(id_cliente =  "' + id +  "');";
}
con.ActualizarDatos(myQuery);
MessageDialog msgBox = new MessageDialog (MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"Los datos han Sido Editados  \n En el Registro ");
msgBox.Show();
ResponseType miResultado = (ResponseType)msgBox.Run ();
msgBox.Destroy();
```

```
if (miResultado == ResponseType.Ok)
                {
                        entry_Id.Text="";
                        entry_Nombre.Text="";
                        entry_apell.Text="";
                        entry_pass.Text="" ;
                        //entry_tipo.Text="";
                        con.LimpiarTeeView();
                        con.cargarTreeView(ref treeviewDatos,Lista,clav);
                }
        }
   }
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs
e)
{
        this.Destroy();
}
protected virtual void OnTreeviewDatosCursorChanged (object sender,
System.EventArgs e)
{
        Gtk.TreeModel model;
        Gtk.TreeIter iter;
        if (treeviewDatos.Selection.GetSelected(out model, out iter)) {
        entry_Id.Text=model.GetValue(iter, 0).ToString();
        entry_Nombre.Text=model.GetValue(iter, 1).ToString();
```

```
            entry_apell.Text=model.GetValue(iter, 2).ToString();
            if(clave=="usuarios")
            {
                  entry_pass.Text=model.GetValue(iter, 3).ToString();
            }
      }
   }
}
}//fin de edición.cs
```

Borrar.cs

```
using System;
using Gtk;
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
      public class Borrar : Gtk.Window
      {
            protected Gtk.Entry entry_id;
            public string id;
            public string cad;
            public string clav;
```

```
ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
protected Gtk.Window MyWin;
public Borrar(string clave ) :
base(Gtk.WindowType.Toplevel)
{
Stetic.Gui.Build(this, typeof(ConectarMySQL.Borrar));
clav=clave;
this.Title="Borrar "+ clave;
//this.DrawRedLine();
}
protected virtual void OnButtonAceptarClicked (object sender,
System.EventArgs e)
{
id= entry_id.Text;
if(clav == "usuarios")
{
cad="DELETE FROM usuarios" +" WHERE (id_usuario = '" +
id + "');";
}
if(clav == "instructor")
{
cad="DELETE FROM instructor" +" WHERE (id_instructor = '" + id +
"');";
}
```

```
if(clav == "cliente")
        {
                cad="DELETE FROM clientes" +" WHERE (id_cliente = '" + id + "');";
        }
                if(entry_id.Text.ToString().Length == 0)
                {
                        MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"El registro no se
puede Eliminar \n Faltan Datos ");
                        msgBox.Show();
                        ResponseType miResultado = (ResponseType)msgBox.Run
();
                        msgBox.Destroy();
                        if (miResultado == ResponseType.Ok){
                        entry_id.Text="";
                }
        }
        else{
                  id=entry_id.Text;
                //id= entry_id.Text;
                con.EliminarDatos(id,cad);
                MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"El Registro ha
sido Eliminado \n Satisfactoriaminte ");
                msgBox.Show();
```

```
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok)
                {
                        entry_id.Text="";
                }
        }
        //id= entry_id.Text;
        //      con.EliminarDatos(id,cad);
    }
        protected virtual void OnButtonCancelarClicked (object sender,
System.EventArgs e)
        {
                this.Destroy();
        }
}//fin de borrar
```

Formulario Instructor.cs

```
using System;
using Gtk;
namespace ConectarMySQL
{
  public class Instructor : Gtk.Window
  {
        protected Gtk.Entry entry_id;
```

```
protected Gtk.Entry entry_nombre;
protected Gtk.Entry entry_apellidos;
protected Gtk.Entry entry_edad;
protected Gtk.CheckButton checkbutton_F;
protected Gtk.CheckButton checkbutton_M;
protected Gtk.Entry entry_Telefono;
protected Gtk.Entry entry_nsala;
protected Gtk.ComboBox cb;
protected Gtk.Window MyWin;
 Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
protected Gtk.Label label1;
string id;
string nomb;
string apell;
string sexo;
string edad;
string cod_clase;
string n_sala;
protected Gtk.Label label_msj;
public Instructor() :
base(Gtk.WindowType.Toplevel)
{
        Stetic.Gui.Build(this, typeof(ConectarMySQL.Instructor));
```

```
label_msj.ModifyFg(Gtk.StateType.Normal,blue_color );
label_msj.ModifyFont(Pango.FontDescription.FromString("Arial Black 12"));
//label1.Text="Pesas";
}
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs e)
{
this.Destroy();
}

protected virtual void OnCheckbuttonMClicked (object sender, System.EventArgs e)
{
sexo="M";
if(checkbutton_F.Active)
checkbutton_F.Active=false;
else
checkbutton_M.Active=true;
}
protected virtual void OnCheckbuttonFClicked (object sender, System.EventArgs e)
{
sexo="F";
if(checkbutton_M.Active)
```

```
                checkbutton_M.Active=false;
                 else
                checkbutton_F.Active=true;
                 //checkbutton_M.Active=false;
        }
protected virtual void OnCbChanged (object sender, System.EventArgs e)
        {
                 TreeIter iter;
                 if (cb.GetActiveIter (out iter))
                cod_clase=((string)cb.Model.GetValue(iter,0));
        }
protected virtual void OnButtonGuardarClicked (object sender, System.EventArgs
e)
        {
                if(entry_id.Text.ToString().Length == 0
||entry_nombre.Text.ToString().Length == 0 ||
entry_apellidos.Text.ToString().Length ==
0||entry_nsala.Text.ToString().Length==0)               {
                        MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"No se puede
Insertar los Datos \n Faltan Datos ");
                        msgBox.Show();
                        ResponseType miResultado = (ResponseType)msgBox.Run
();
                        msgBox.Destroy();
                        if (miResultado == ResponseType.Ok){
```

```
                    entry_id.Text="";
                    entry_nombre.Text="";
                    entry_apellidos.Text="";
                    entry_edad.Text="" ;
                    entry_Telefono.Text="";
                    entry_nsala.Text="";
                    //entry_tipo.Text="";
                    }
                }
                else{
                    id = this.entry_id.Text;
                    nomb= this.entry_nombre.Text;
                    apell=this.entry_apellidos.Text;
                    edad= this.entry_edad.Text;
                     n_sala=this.entry_nsala.Text;
                      if(con.EscribirDatosInstructor(id,nomb,apell,edad,sexo))
                     con.EscribirDatosClase(cod_clase,id,n_sala);
                      MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"Los datos han
sido Insertados  \n En el Registro ");
                    msgBox.Show();
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                    msgBox.Destroy();
```

```
if (miResultado == ResponseType.Ok)
                        {
                                entry_id.Text="";
                                entry_nombre.Text="";
                                entry_apellidos.Text="";
                                entry_edad.Text="" ;
                }
        }
```

Formulario Cliente.cs

```
using System;
using Gtk;
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
    public class Cliente : Gtk.Window
        {
                protected Gtk.Entry entry_id;
                protected Gtk.Entry entry_nomb;
                protected Gtk.Entry entry_edad;
                protected Gtk.CheckButton chkb1_M;
                protected Gtk.CheckButton chkb_F;
                Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
```

```
protected Gtk.Window MyWin;
ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
string id;
string nomb;
string apell;
string sexo;
string edad;
string cod_clase="";
protected Gtk.Entry entry_apell;
protected Gtk.ComboBox cb_Clase;
protected Gtk.Label label_msj;
public Cliente() :
base(Gtk.WindowType.Toplevel)
{
Stetic.Gui.Build(this, typeof(ConectarMySQL.Cliente));
label_msj.ModifyFont(Pango.FontDescription.FromString("Arial black 14"));
label_msj.ModifyFg(Gtk.StateType.Normal,blue_color);
}
protected virtual void OnButtonAceptarClicked (object sender, System.EventArgs e)
{
if(entry_id.Text.ToString().Length == 0 ||entry_nomb.Text.ToString().Length == 0 || entry_apell.Text.ToString().Length == 0||cod_clase.Length==0)
```

```
                {
                        MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"No se puede
Insertar los Datos \n Faltan Datos ");
                        msgBox.Show();
                        ResponseType miResultado = (ResponseType)msgBox.Run
();
                        msgBox.Destroy();
                        if (miResultado == ResponseType.Ok){
                        entry_id.Text="";
                        entry_nomb.Text="";
                        entry_apell.Text="";
                        entry_edad.Text="" ;
                        }
        }
        else{
                id = this.entry_id.Text;
                nomb= this.entry_nomb.Text;
                apell=this.entry_apell.Text;
                edad= this.entry_edad.Text;
                 // n_sala=this.entry_.Text;
                if(con.EscribirDatosCliente(id,nomb,apell,edad,sexo))
                con.EscribirDatosAsiste(id,cod_clase);
```

```
                MessageDialog msgBox = new MessageDialog
        (MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"Los datos
han Sido Insertados  \n En el Registro ");
                msgBox.Show();
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok)
                {
                        entry_id.Text="";
                        entry_nomb.Text="";
                        entry_apell.Text="";
                        entry_edad.Text="" ;
                        //entry_Telefono.Text="";
                        //entry_nsala.Text="";
                }
        }
   }
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs
e)
        {
                this.Destroy();
        }
protected virtual void OnCbClaseChanged (object sender, System.EventArgs e)
        {
                TreeIter iter;
```

```
            if (cb_Clase.GetActiveIter (out iter))
            cod_clase=((string)cb_Clase.Model.GetValue(iter,0));
        }
protected virtual void OnChkb1MClicked (object sender, System.EventArgs e)
        {
            sexo="F";
            if(chkb1_M.Active)
            chkb1_M.Active=false;
            else
            this.chkb_F.Active=true;
        }
    }
```

Formulario Sala.cs

```
using System;
using Gtk;
// Se agregan las referencias necesarias para conectar a la DB
using System.Data;
using MySql.Data.MySqlClient;
namespace ConectarMySQL
{
public class Sala : Gtk.Window
{
        protected Gtk.Entry entryno_sala;
```

```
protected Gtk.Entry entrycapsala;
protected Gtk.ComboBox cb_sala;
protected Gtk.Label label1;
Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
public ConectarMySQL.Conexion con= new ConectarMySQL.Conexion();
string nSala;
string tipoSala="";
string capSala;
int csala;
protected Gtk.Entry entrySeleccionado;
protected Gtk.Window MyWin;
public Sala() : base(Gtk.WindowType.Toplevel)
{
        Stetic.Gui.Build(this, typeof(ConectarMySQL.Sala));
        label1.ModifyFont(Pango.FontDescription.FromString("Arial black 14"));
        label1.ModifyFg(Gtk.StateType.Normal,blue_color);
}
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs e)
{
        this.Destroy();
}
```

```
protected virtual void OnButtonAceptarClicked (object sender, System.EventArgs
e)
        {
            if(entryno_sala.Text.ToString().Length == 0
||entrycapsala.Text.ToString().Length == 0 || tipoSala.Length==0)
                {
                MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"No se puede
Insertar los Datos \n Faltan Datos ");
                msgBox.Show();
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){
                entryno_sala.Text="";
                entrycapsala.Text="";
                }
         }
        else{
                nSala=entryno_sala.Text;
                capSala=entrycapsala.Text;
                csala=System.Int32.Parse(capSala);
                con.EscribirDatosSala(nSala,tipoSala,csala);
                 MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"Los datos han
Sido Insertados  \n En el Registro ");
msgBox.Show();
```

```
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok)
                {
                        entryno_sala.Text="";
                        entrycapsala.Text="";
                }
            }
        }
        protected virtual void OnCombobox1Changed (object sender,
System.EventArgs e)
        {
                TreeIter iter;
                if (cb_sala.GetActiveIter (out iter))
                tipoSala=((string)cb_sala.Model.GetValue(iter,0));
        }
  }
}//fin sala.cs
```

Formulario Aparatos.cs

```
using System;
using Gtk;
namespace ConectarMySQL
{
public class Aparatos : Gtk.Window
```

```
{
        protected Gtk.Label label1;
        protected Gtk.Label label2;
        protected Gtk.TreeView treeviewSalas;
        protected Gtk.Label label3;
        protected Gtk.TreeView treeviewAparatos;
        public string myQuery;
        public string myQuery1;
        public string codigo;
        public string cod_aparato="";
        protected Gtk.Window MyWin;
         ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
        protected Gtk.Button btn_Nuevo;
        protected Gtk.Button buttonBorrar;
        protected Gtk.ScrolledWindow scrolledwindow1;
        protected Gtk.ScrolledWindow scrolledwindow2;
        Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
         public Aparatos() : base(Gtk.WindowType.Toplevel)
        {
                this.SetDefaultSize(350,400);
                myQuery= "SELECT * FROM " + "sala";
                 Stetic.Gui.Build(this, typeof(ConectarMySQL.Aparatos));
            treeviewSalas.AppendColumn("Numero de sala", new
Gtk.CellRendererText(), "text", 0);
```

```
                treeviewSalas.AppendColumn("Tipo de sala ", new
Gtk.CellRendererText(), "text", 1);
                treeviewSalas.AppendColumn("Capasidad de sala", new
Gtk.CellRendererText(), "text", 2);
                //Campos de aparatos
                treeviewAparatos.AppendColumn("Codigo del Aparato", new
Gtk.CellRendererText(), "text", 0);
                treeviewAparatos.AppendColumn("Nombre del Aparato ", new
Gtk.CellRendererText(), "text", 1);
                treeviewAparatos.AppendColumn("Vida Util", new
Gtk.CellRendererText(), "text", 2);
                //cargar las Salas
                label1.ModifyFg(Gtk.StateType.Normal,blue_color );
                label1.ModifyFont(Pango.FontDescription.FromString("Arial
Black 14"));
                label2.ModifyFg(Gtk.StateType.Normal,blue_color );
                label2.ModifyFont(Pango.FontDescription.FromString("Arial
Black 14"));
                con.cargarTreeViewSala(ref treeviewSalas,myQuery);
        }
    protected virtual void OnBtnCerrarClicked (object sender,
System.EventArgs e)
        {
            this.Destroy();
        }
```

```
protected virtual void OnTreeviewSalasCursorChanged (object sender,
System.EventArgs e)
		{
			Gtk.TreeModel model;
			Gtk.TreeIter iter;
			if (treeviewSalas.Selection.GetSelected(out model, out iter))
			{
				codigo =""+ model.GetValue(iter, 0).ToString();
				//Console.WriteLine(model.GetValue(iter, 1).ToString());
				this.btn_Nuevo.Show();
				this.buttonBorrar.Show();
				this.treeviewAparatos.Show();
			}
			myQuery1= "SELECT * FROM  aparatos WHERE (no_sala like
'"+ codigo + "')";
			con.cargarTreeViewAparato(ref treeviewAparatos,myQuery1);
		}
	protected virtual void OnBtnNuevoClicked (object sender,
System.EventArgs e)
	{
		ConectarMySQL.Nuevo_aparato nAparato = new
Nuevo_aparato(código);
		nAparato.Show();
	}
```

```
protected virtual void OnButtonBorrarClicked (object sender,
System.EventArgs e)
{
if (cod_aparato.Length == 0)
{
MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"Debe
Selecccionar un  Aparato \n de la sala");
msgBox.Show();
ResponseType miResultado = (ResponseType)msgBox.Run ();
msgBox.Destroy();
if (miResultado == ResponseType.Ok){
con.cargarTreeViewAparato(ref treeviewAparatos,myQuery1);
}
}
else {
if(con.EliminarAparato(cod_aparato)){
MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Info,ButtonsType.Ok,"El Aparato se
Elimino\n de la sala" +codigo );
msgBox.Show();
ResponseType miResultado = (ResponseType)msgBox.Run ();
msgBox.Destroy();
if (miResultado == ResponseType.Ok){
con.cargarTreeViewAparato(ref treeviewAparatos,myQuery1);
```

```
                    }
                }
            }
        }
protected virtual void OnTreeviewAparatosCursorChanged (object sender,
System.EventArgs e)
        {
            Gtk.TreeModel model;
            Gtk.TreeIter iter;
            if (treeviewAparatos.Selection.GetSelected(out model, out iter))
            {
                cod_aparato ="" + model.GetValue(iter, 0).ToString();
                //Console.WriteLine(model.GetValue(iter, 0).ToString()); }
        }
} //fin de Aparatos.cs
```

Nuevo_aparato.cs

```
using System;
using Gtk;
namespace ConectarMySQL
{
public class Nuevo_aparato : Gtk.Window
{
        protected Gtk.Entry entry_codigo;
        protected Gtk.Entry entry_nombre;
```

```
protected Gtk.Entry entry_vida_util;
protected Gtk.Entry entry_n_sala;
Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
protected Gtk.Window MyWin;
string cod;
string nombre;
string v_util;
string nsala;
int vutil;
protected Gtk.Label label1;
public Nuevo_aparato(string cod_sala): base(Gtk.WindowType.Toplevel)
{
        Stetic.Gui.Build(this, typeof(ConectarMySQL.Nuevo_aparato));
        label1.ModifyFont(Pango.FontDescription.FromString("Arial black
14"));
        label1.ModifyFg(Gtk.StateType.Normal,blue_color);
        nsala=cod_sala;
         entry_n_sala.Text=nsala;
}
protected virtual void OnButtonCancelarClicked (object sender, System.EventArgs
e)
{
        this.Destroy();
}
```

```
protected virtual void OnButtonAceptarClicked (object sender, System.EventArgs
e)
        {
        if(entry_codigo.Text.ToString().Length == 0
||entry_nombre.Text.ToString().Length == 0 ||
entry_vida_util.Text.ToString().Length == 0||entry_n_sala.Text.ToString().Length
== 0)
        {
                MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Aparato no se
puede Añadirse en la sala "+ nsala  +" \n \tFaltan Datos ");
                ResponseType miResultado = (ResponseType)msgBox.Run ();
                msgBox.Destroy();
                if (miResultado == ResponseType.Ok){
                entry_codigo.Text="";
                entry_nombre.Text="";
                entry_vida_util.Text="";
                }
        }
        else{
                cod=entry_codigo.Text;
                nombre=entry_nombre.Text;
                v_util=entry_vida_util.Text;
                vutil= System.Int32.Parse(v_util);
```

```
if( con.EscribirDatosaAparatos(cod,nombre,vutil,nsala))
                {
                        MessageDialog msgBox = new MessageDialog
(MyWin,DialogFlags.Modal,MessageType.Error,ButtonsType.Ok,"El Usuario ha
sido Añadido \n En el Registro ");
                        msgBox.Show();
                        ResponseType miResultado = (ResponseType)msgBox.Run
();
                        msgBox.Destroy();
                        if (miResultado == ResponseType.Ok)
                        {
                                entry_codigo.Text="";
                                entry_nombre.Text="";
                                entry_vida_util.Text="";
                        }
                }
                else {
                        entry_codigo.Text="";
                        entry_nombre.Text="";
                        entry_vida_util.Text="";
                        }
                }
        }
}
}//fin de Nuevo_aparato.cs
```

Formulario ListaClientes.cs

```
using System;
namespace ConectarMySQL
{
public class ListaClientes : Gtk.Window
{
        protected Gtk.TreeView treeview1;
        protected Gtk.Label label1;
        protected Gtk.ScrolledWindow scrolledwindow1;
        public string myQuery;
        ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
        Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
        public ListaClientes(string clase) : base(Gtk.WindowType.Toplevel)
        {
                Stetic.Gui.Build(this, typeof(ConectarMySQL.ListaClientes));
                treeview1.AppendColumn("ID", new Gtk.CellRendererText(), "text",
0);
                treeview1.AppendColumn("Nombre", new Gtk.CellRendererText(),
"text", 1);
                treeview1.AppendColumn("Apellidos", new Gtk.CellRendererText(),
"text", 2);
                treeview1.AppendColumn("Edad", new Gtk.CellRendererText(),
"text", 3);
                treeview1.AppendColumn("Sexo", new Gtk.CellRendererText(),
"text", 4);
```

```
                treeview1.AppendColumn("Clase", new Gtk.CellRendererText(),
"text", 5);
                label1.ModifyFg(Gtk.StateType.Normal,blue_color );
                label1.ModifyFont(Pango.FontDescription.FromString("Arial Black
14"));
                label1.Text=label1.Text +" " + clase;

        scrolledwindow1.SetSizeRequest(250,200);//.SetSizeRequest(200,300);
                treeview1.Columns[3].Visible=false;
                this.SetDefaultSize(400,400);
                myQuery = "SELECT * FROM "+ "clientes,asiste WHERE
"+"clientes.id_cliente = asiste.id_cliente and cod_clase= " + "'"+ clase +"'";
                con.cargarTreeViewCliente(ref treeview1,myQuery);
                //myQuery= "SELECT * FROM " + "instructor";
        }
        protected virtual void OnBtnCerrarClicked (object sender,
System.EventArgs e)
        {
                this.Destroy();
        }
       }
}//fin de  ListaClientes.cs
```

Formulario Listinst.cs

```
using System;
namespace ConectarMySQL
{
	public class ListInst : Gtk.Window
	{
		protected Gtk.TreeView treeview1;
		protected Gtk.ScrolledWindow scrolledwindow1;
		public string myQuery;
		ConectarMySQL.Conexion con =new ConectarMySQL.Conexion();
		protected Gtk.Label label1;
		Gdk.Color blue_color = new Gdk.Color (0, 0, 0xff);
		public ListInst() : base(Gtk.WindowType.Toplevel)
		{
		Stetic.Gui.Build(this, typeof(ConectarMySQL.ListInst));
		treeview1.AppendColumn("ID", new Gtk.CellRendererText(), "text",
0);
		treeview1.AppendColumn("Nombre", new Gtk.CellRendererText(),
"text", 1);
		treeview1.AppendColumn("Apellidos", new Gtk.CellRendererText(),
"text", 2);
		treeview1.AppendColumn("Edad", new Gtk.CellRendererText(),
"text", 3);
		treeview1.AppendColumn("Sexo", new Gtk.CellRendererText(),
"text", 4);
```

```
                treeview1.AppendColumn("Telefono", new Gtk.CellRendererText(),
"text", 5);
                label1.ModifyFg(Gtk.StateType.Normal,blue_color );
                label1.ModifyFont(Pango.FontDescription.FromString("Arial Black
14"));

        scrolledwindow1.SetSizeRequest(250,200);//.SetSizeRequest(200,300);
                treeview1.Columns[3].Visible=false;
                this.SetDefaultSize(400,400);
                myQuery= "SELECT * FROM " + "instructor";
                con.cargarTreeViewInst(ref treeview1,myQuery);
                }
        protected virtual void OnBtnCerrarClicked (object sender,
System.EventArgs e)
                {
                 this.Destroy();
                }
        }
}//fin de Listinst.cs
```

Formulario Ayuda.cs

```
using Gtk;
using System;
using System.Reflection;
namespace ConectarMySQL
```

```
{
        public class Ayuda
        {
        public Ayuda()
        {
       // Application.Init ();
        AboutDialog dialog = new AboutDialog ();
        Assembly asm = Assembly.GetExecutingAssembly ();
       dialog.Name =(asm.GetCustomAttributes (
       typeof (AssemblyTitleAttribute), false) [0]
       as AssemblyTitleAttribute).Title;
        dialog.Version = "Gimnasio Acción Club \n Versión 1.0 ";
        dialog.Comments = (asm.GetCustomAttributes ( typeof
(AssemblyDescriptionAttribute), false) [0] as
AssemblyDescriptionAttribute).Description;
         dialog.Copyright = (asm.GetCustomAttributes (
       typeof (AssemblyCopyrightAttribute), false) [0] as
AssemblyCopyrightAttribute).Copyright;
        dialog.License = license;
        dialog.Authors = authors;
         //dialog.AddButton("Cerrar",ResponseType.Close);
         dialog.Run ();
         dialog.Destroy();
    }
  private static string [] authors = new string [] {
```

```
        "Marlin José Rivera S. <rivera.marlin@gmail.com >",
        "Meyling Yuniett Silva R.<meylingsilvarivera@gmail.com>",
        "Francisco José Silva S. <fco_silva2000@yahoo.es>"
    };
 private static string license =
@"Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
""Software""), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.
the software is provided ""as is"", without warranty of any kind,
express or implied, including but not limited to the warranties of
merchantability, fitness for a particular purpose and
noninfringement. in no event shall the authors or copyright holders be
liable for any claim, damages or other liability, whether in an action
of contract, tort or otherwise, arising from, out of or in connection
with the software or the use or other dealings in the software.";
  }
}//fin ayuda.cs
```

XIII. Conclusiones

En el transcurso de nuestro trabajo monográfico hemos podido comprobar mediante un ejemplo útil que Monodevelop es una herramienta muy potente y fácil de utilizar, además, ha llenado nuestras expectativas propuestas como desarrolladores de sistemas, tanto en la velocidad de creación de interfaces gráficas como en su portabilidad, así como también nos ha permitido lograr nuestros objetivos propuestos.

Al utilizar MySQL como gestor de bases de datos obtuvimos grandes facilidades, ya que posee herramientas gráficas que nos permitieron la rápida creación de nuestra base de datos (gimnasio) como son MySQL Administrador y MySQL Query Browser.

Cuando utilizamos la herramienta Monodevelop para la creación de nuestra aplicación obtuvimos muchas facilidades y al compararlo con otros diseñadores de interfaz gráfica, podemos afirmar que poseen muchas características similares, pero las aplicaciones desarrolladas con nuestra herramienta son portables y pueden utilizar licencia GPL.

Los inconvenientes que se nos presentaron en la elaboración de este trabajo fueron referentes al desconocimiento de las librerías que posee Monodevelop pero al obtener la información necesaria de las mismas los pudimos superar fácilmente, gracias a la amplia comunidad DevelopArts en Internet.

La trilogía Monodevelop, MySQL y OpenSuse nos ha demostrado que se pueden crear aplicaciones grandes y complejas en poco tiempo y con pocos recursos en comparación de otros entornos de desarrollo.

XIV. Recomendaciones

Después de la exposición de nuestras conclusiones podemos recomendar que:

- El Departamento de Computación de la Universidad Nacional Autónoma de Nicaragua (UNAN-LEÓN) incorpore en los componentes curriculares la programación de aplicaciones graficas en .NET con software libre.

- Para la realización de futuros proyectos más complejos, se puede utilizar la herramienta de desarrollo Visual Studio de Microsoft Windows e implementarla en Monodevelop de Linux.

- Instalar las versiones mas recientes de los programas descritos en nuestro trabajo monográfico para el correcto funcionamiento de los mismos.

- La lectura de la documentación adjunta a la instalación de Monodevelop.

- El uso de Internet y de la comunidad DevelopArts, se debe hacer para la obtención de información extra, pues, esta servirá en la solución de posibles problemas con el uso de la herramienta Monodevelop.

- Utilización de software libre para evitar problemas legales.

- Tomar este trabajo monográfico como referencia para crear futuras aplicaciones

XV. Bibliografía

Libros:

- Ceballos Francisco Javier (2006) “Enciclopedia de Microsoft Visual C#”, México, ALFAOMEGA.
- Adoración de Miguel Castaño, Mario Piattini Velthuis, Esperanza Marcos Martínez (1999) “Diseño de bases de datos relacionales”, España, Rama.
- Manual de referencia de MySQL 5.0,
- Páginas del manual de Linux (MAN).

Referencias de Internet:

- **http://www.DevelopArts.com**
- **http://www.mysql.com/**
- **http://www.traductoronline.com/**
- **http://www.Opensuse.org/**

XVI. Anexos

16.1 Introducción a OpenSUSE:

El proyecto OpenSUSE es un programa de y para la comunidad, patrocinado por Novell. Al tiempo que promueve el uso de Linux, OpenSUSE.org proporciona un acceso libre y sencillo a la distribución de Linux número uno en facilidad de uso, SUSE Linux. OpenSUSE ofrece a desarrolladores y usuarios todo lo que necesitan para empezar a trabajar con Linux. Los objetivos del proyecto openSUSE son:

- Convertir a SUSE Linux en la distribución Linux más fácil de obtener para cualquier persona y en la plataforma de código abierto más extendida.

- Crear un entorno de colaboración en la comunidad de código abierto para hacer de SUSE Linux la mejor distribución a escala mundial para usuarios con experiencia y recién llegados al mundo de Linux.

- Simplificar y abrir los procesos de desarrollo y construcción de paquetes con el fin de que SUSE Linux sea la primera elección como plataforma para programadores de Linux y desarrolladores de aplicaciones.

16.2 Instalación de OpenSuse

SUSE Linux 10.3

Booteamos en CD, DVD, o el medio que hayamos elegido para instalar y nos vamos encontrar con la imagen de Bienvenida, signo de buena educación, y las opciones de instalación, idioma, kernel, fuente de instalación, resolución, etc.

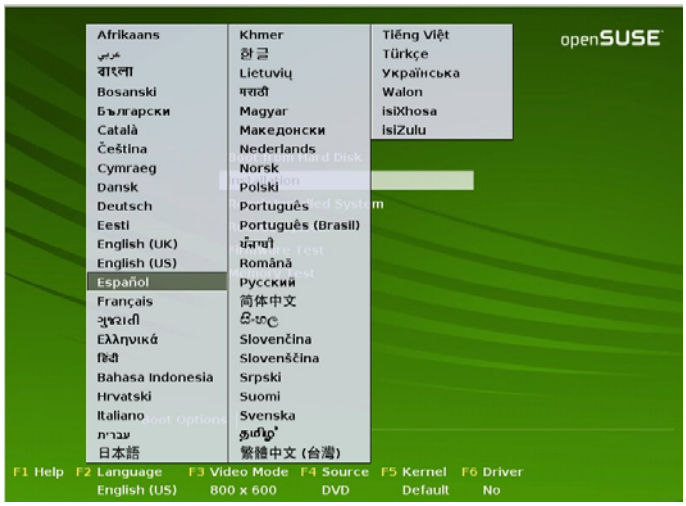

Fig. 42 Cambio de Idioma

Ya elegidas las opciones, vamos a la Instalación y damos enter, aquí va ir cargando el kernel.

Fig. 43 Cargando el kernel

Ya cargado el kernel, nos da la opción de hacer una comprobación del medio de instalación, según el caso, si sabemos que esta bien nuestro disco podemos saltarnos la comprobación, en caso de tener alguna duda es muy recomendable que hagamos la prueba... La comprobación de medio no demora más que unos minutos.

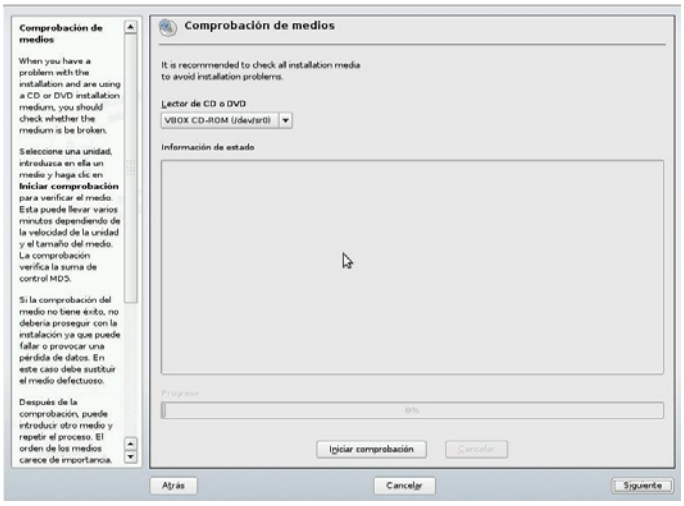

Fig. 44 Comprobación del medio de instalación

Ahora le damos clic en el botón de Radio SI, al acuerdo de licencia y seguimos

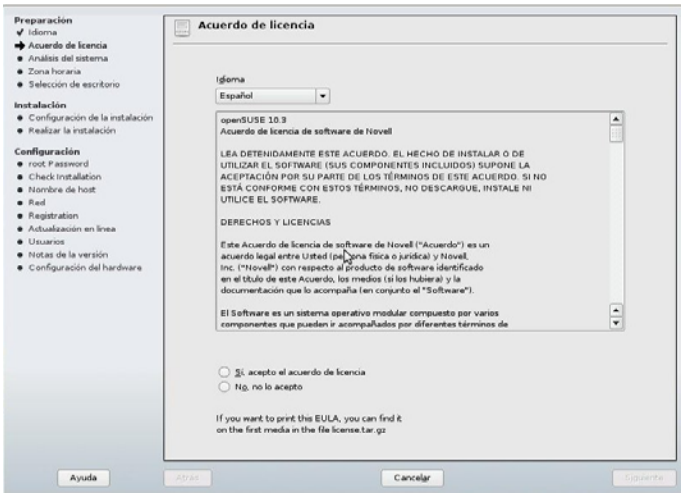

Fig. 45 Acuerdo de licencia

El instalador hace una detección de medios, dispositivos, etc.

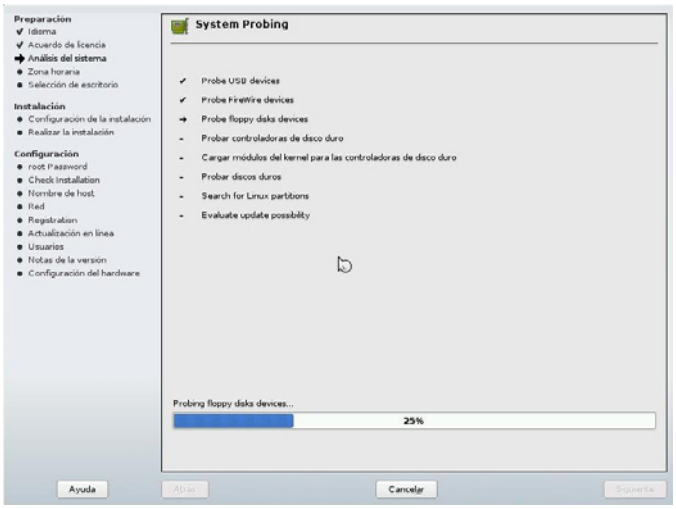

Fig. 46 Detección de medios

Una vez hecha la prueba de medios, damos a siguiente y nos va a preguntar el modo de instalación, ponemos Nueva Instalación que es lo queremos, en este punto también podemos actualizar un sistema Suse instalado previamente y/o reparar una instalación con problemas.

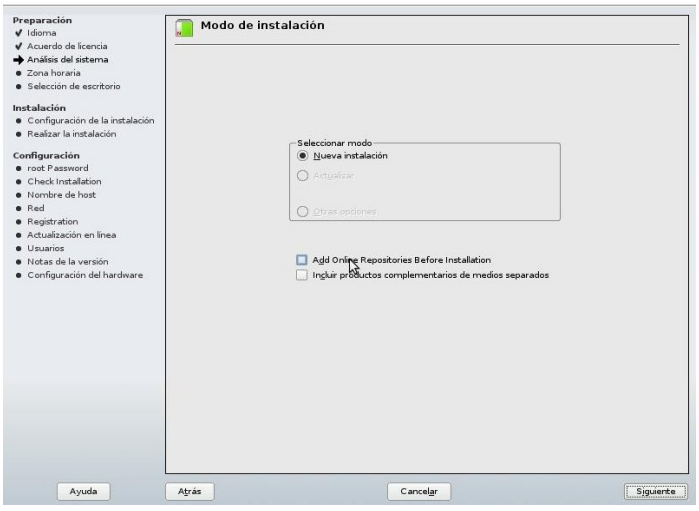

Fig. 47 Modo de Instalación

Podemos ver además el clásico panel de instalación de Suse, con su panel que nos va guiando durante la instalación, muy útil. Clic en siguiente y ahora hace un refresco de los repositorios.

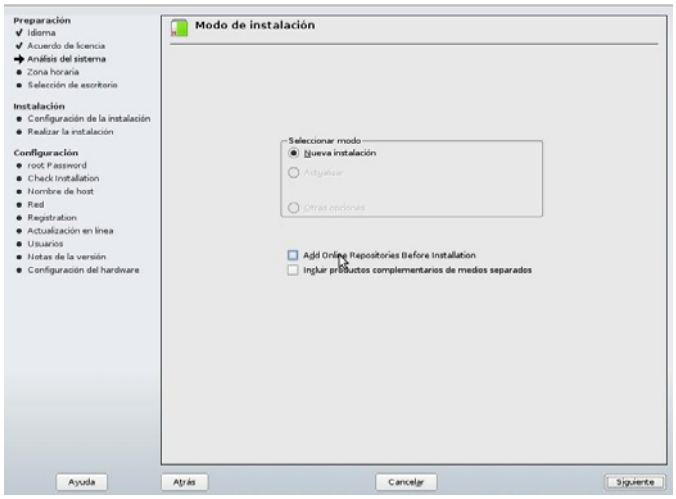

Fig. 48 Modo de Instalación

Datos de configuración. Ahora nos va a ir pidiendo una serie de datos para configurar el sistema estos son: Reloj y Zona Horaria.

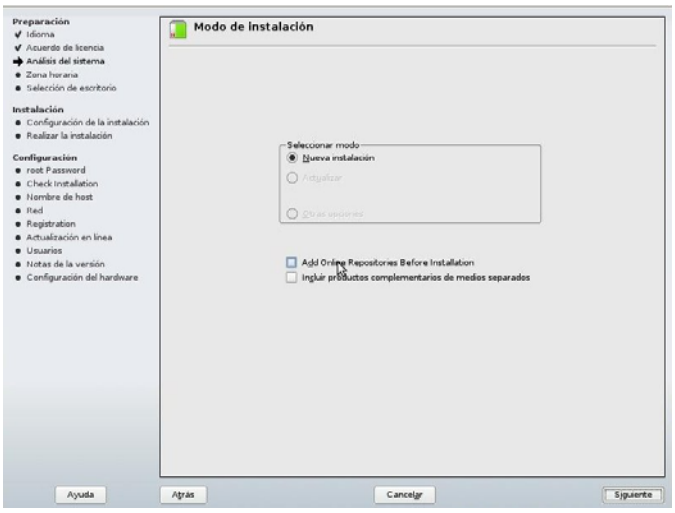

Fig. 49 Modo de Instalación

Selección de Escritorio

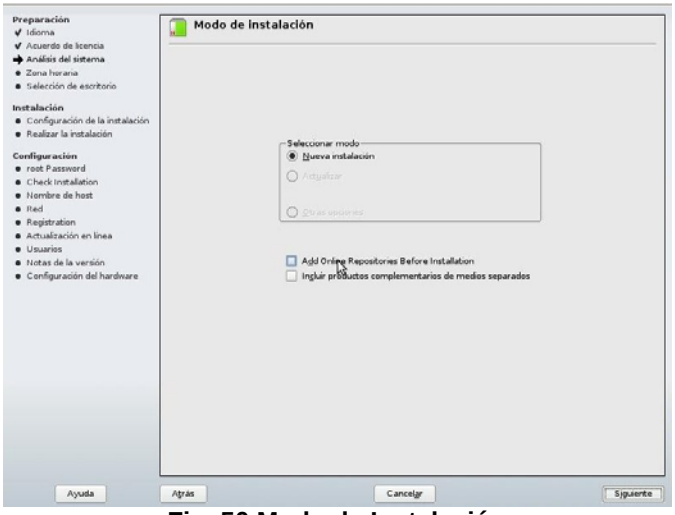

Fig. 50 Modo de Instalación

Resumen de Instalación

Ya con los datos básicos, ahora nos muestra un resumen con las configuraciones de Instalación, acá tenemos que ver si va a instalar lo que queremos, en donde queremos, si están bien los datos, etc, para esto podemos ver cada uno de los ítems

- Particionamiento
- Software
- Configuración Regional

Si hay algún dato que no este bien o no sea como queremos, simplemente lo cambiamos haciendo clic en la opción, es recomendable ver bien las particiones en caso de tener antes algún otro sistema operativo o particiones previas, otro detalle importante es el lugar donde nos va a instalar el gestor de arranque y cual va a ser, grub o lilo, son los dos con que trabaja OpenSuse.

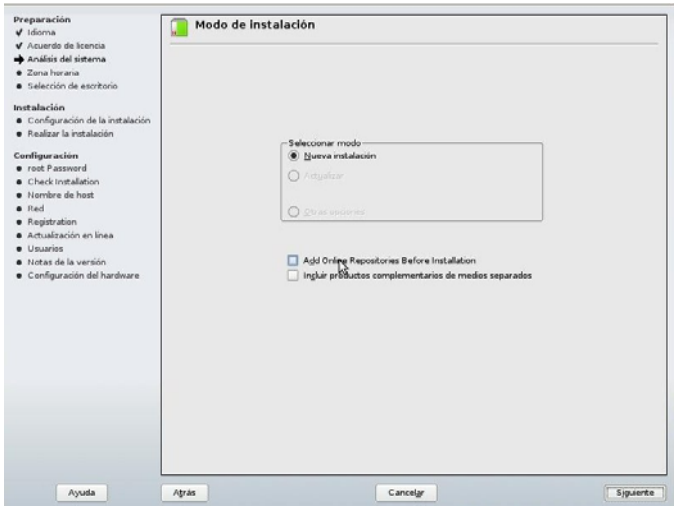

Fig. 51 Modo de Instalación

Es aquí donde definimos los detalles de la instalación, ya todo revisado y bien, le damos a Siguiente. Ahora nos va a pedir confirmar unos acuerdos de licencia para los productos Adobe.

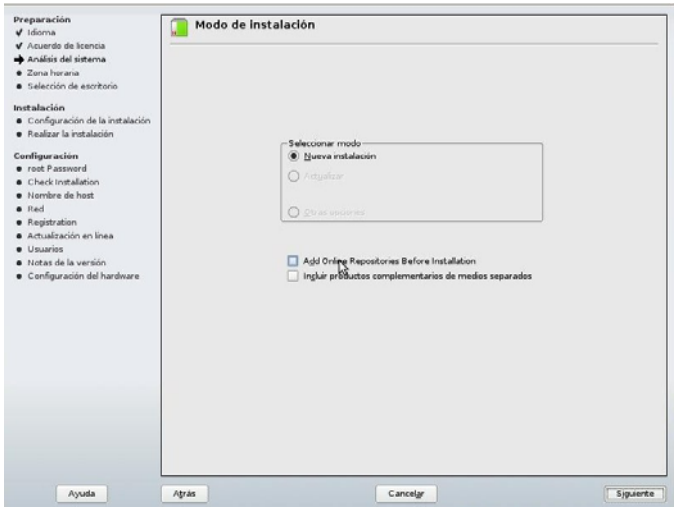

Fig. 52 Modo de Instalación

Confirmamos Instalación

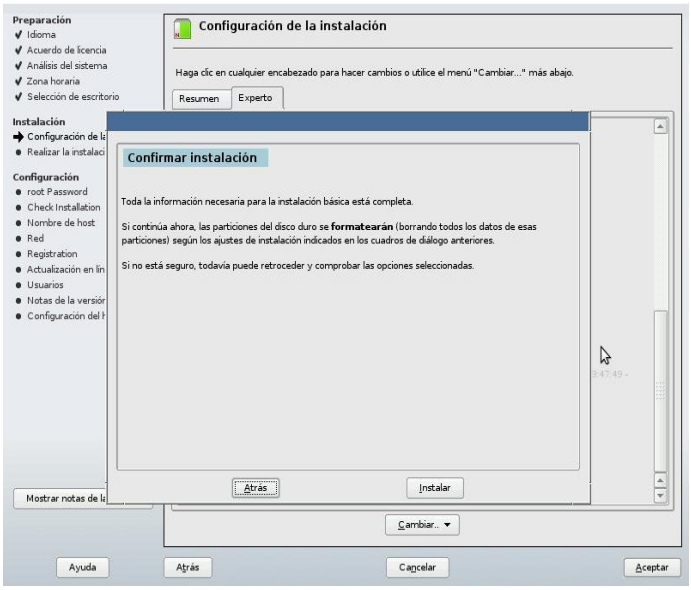

Fig. 53 Configuración de Instalación

Realizando la Instalación

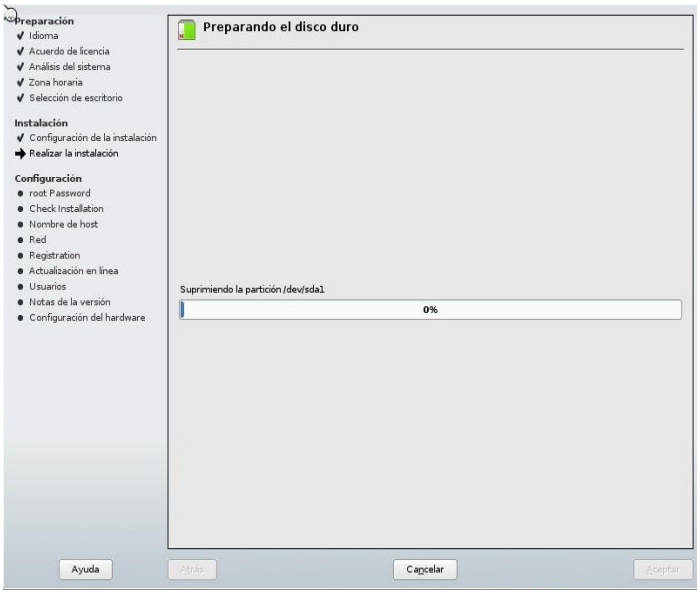

Fig. 54 Preparando el disco duro

Instalando los paquetes que seleccionamos antes

Fig. 55 Instalación de paquetes

La misma pantalla, pero mostrándonos los detalles de instalación de paquetes

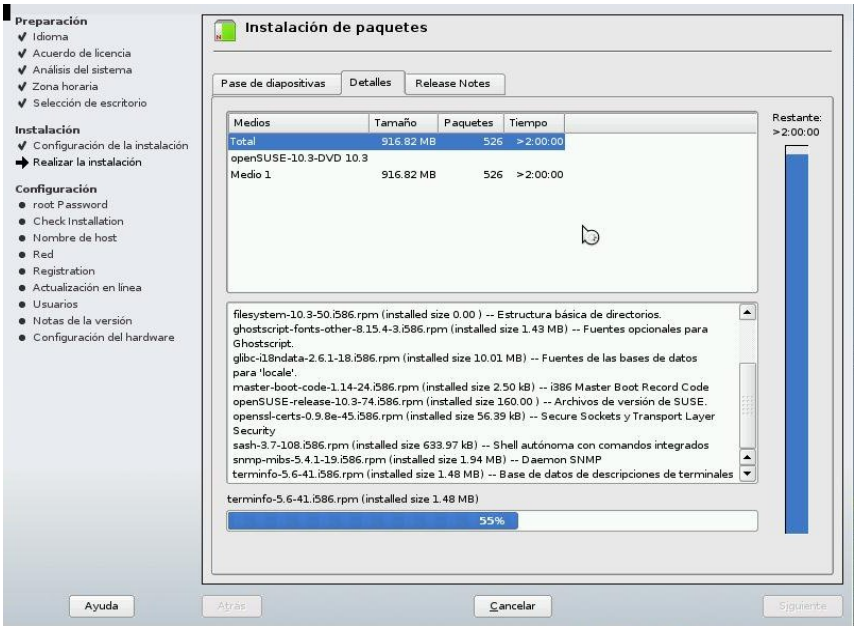

Fig. 56 Instalación de paquetes

Terminando la Instalación Básica

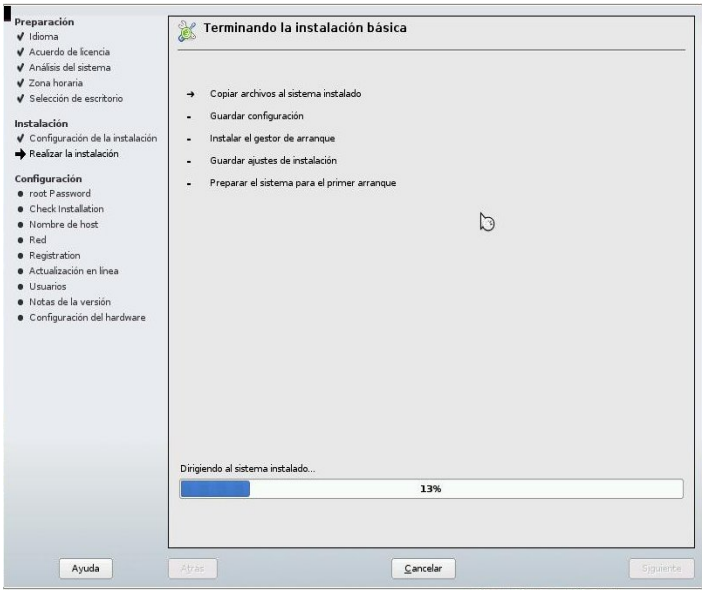

Fig. 57 Terminando instalación básica

Una vez que ya termino la instalación básica, es la hora del primer reinicio

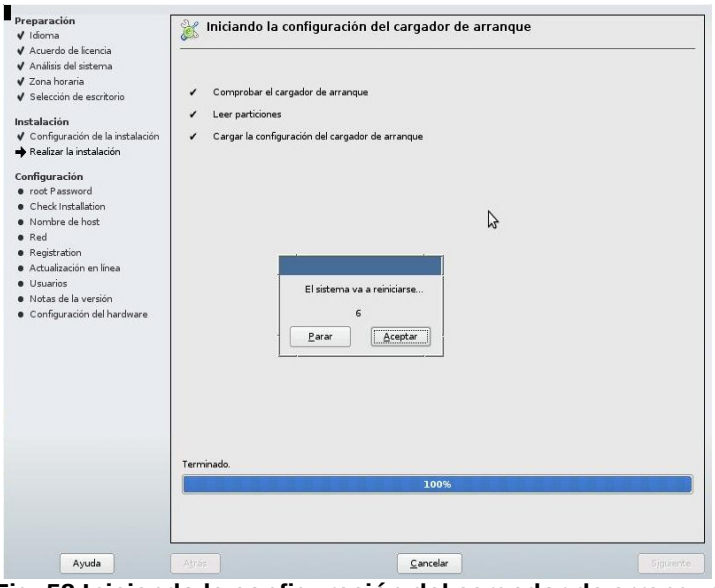

Fig. 58 Iniciando la configuración del cargador de arranque

Configuraciones: De vuelta del reinicio, entramos ya a nuestro sistema OpenSuse instalado en el disco duro y terminamos de configurar los últimos detalles: Contraseña de Root.

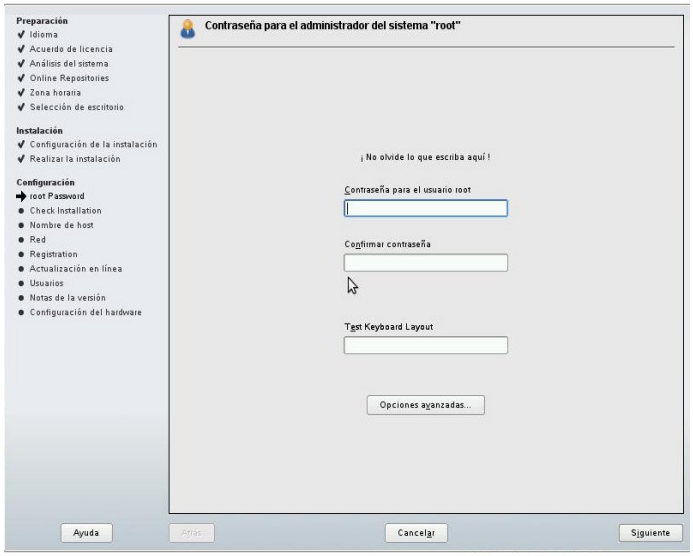

Fig. 59 Contraseña para el administrador del sistema

Nombre de host y de Dominio

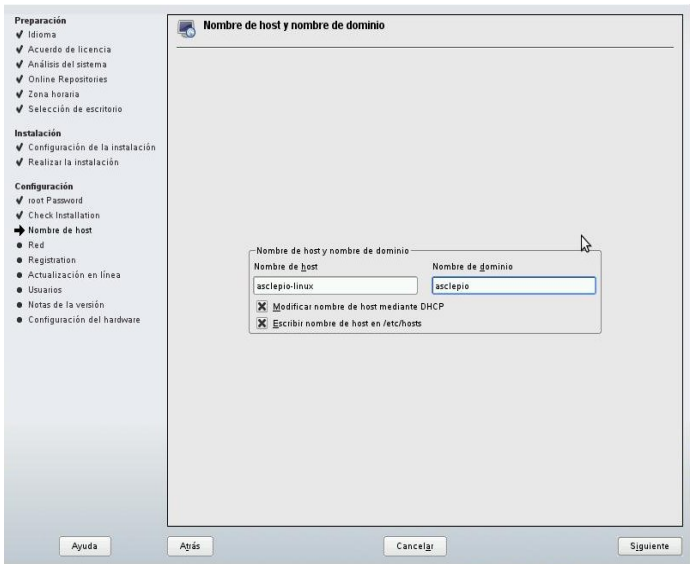

Fig. 60 Nombre del host y nombre del dominio

Configuramos la red

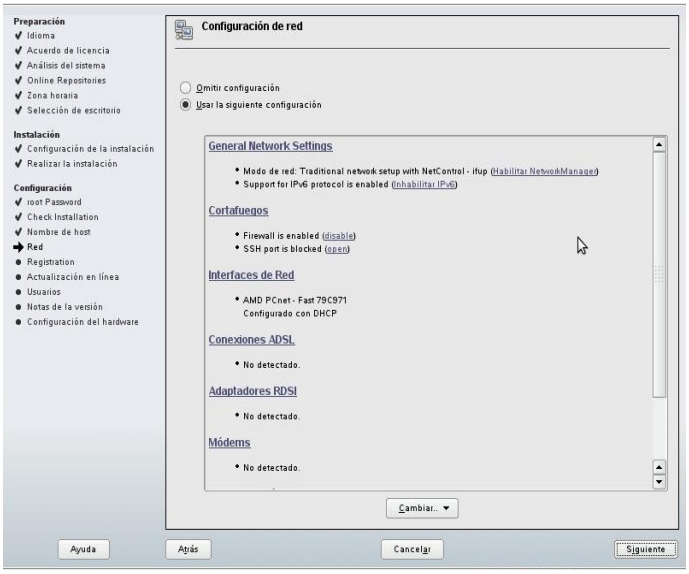

Fig. 61 Configuración de red

Guarda los datos de configuración

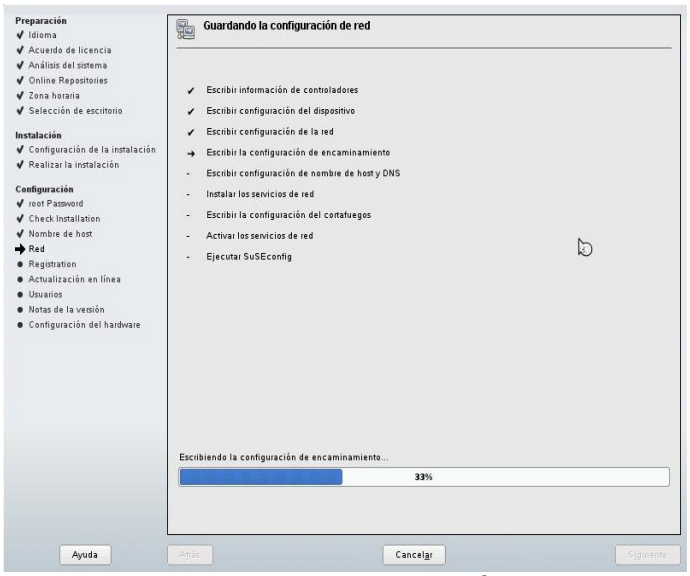

Fig. 62 Guardando configuración de red

Nos pregunta si queremos probar la conexión a Internet, esto es algo poco relevante porque lo podemos hacer después ya con las Xs corriendo (entorno gráfico), le ponemos que no y seguir.

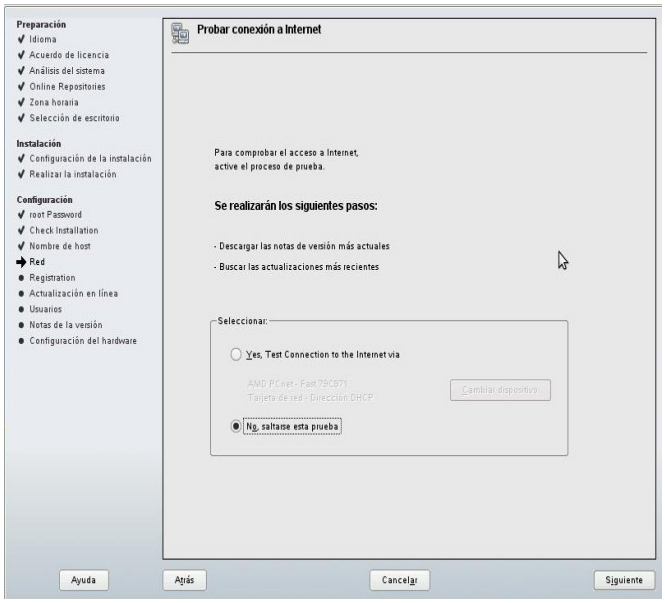

Fig. 63 Probar conexión a Internet

Método de Autentificación de Usuario, le dejamos por defecto.

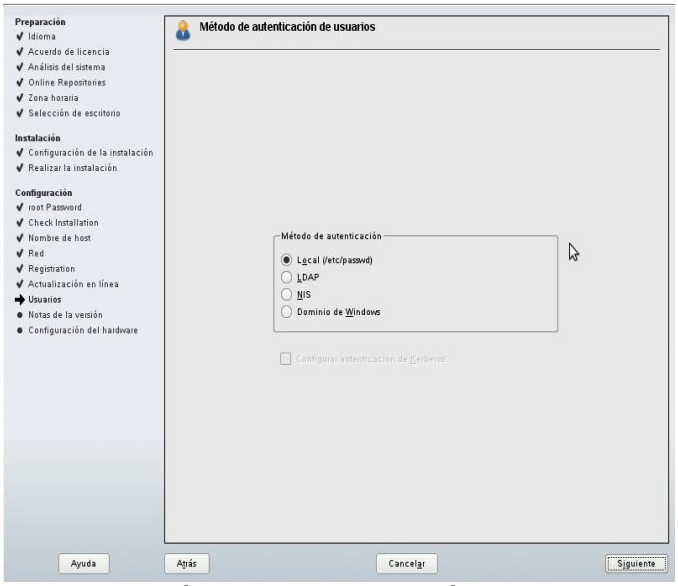

Fig. 64 Método de autentificación de usuario

Hechas las configuraciones, ahora el instalador las escribe en el sistema

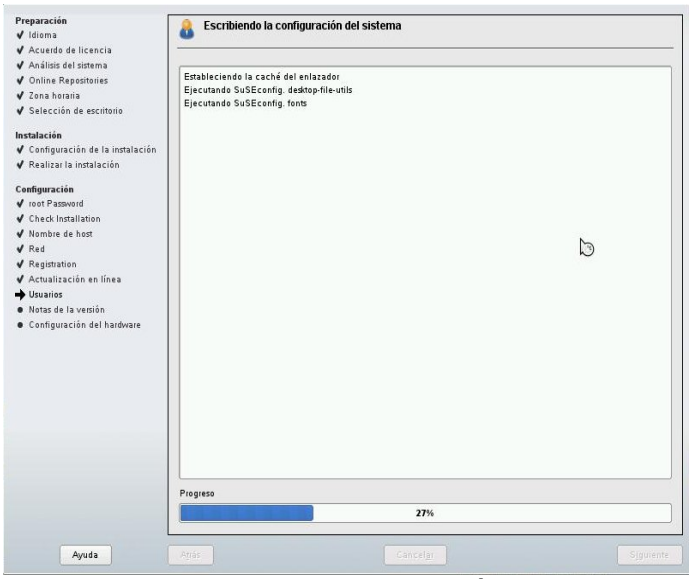

Fig. 65 Escribiendo configuración de usuario

Notas de versión, básicamente es un resumen de la versión de OpenSuse y cosas que nunca leemos y esta vez tampoco va a ser la excepción, así que le damos a Siguiente.

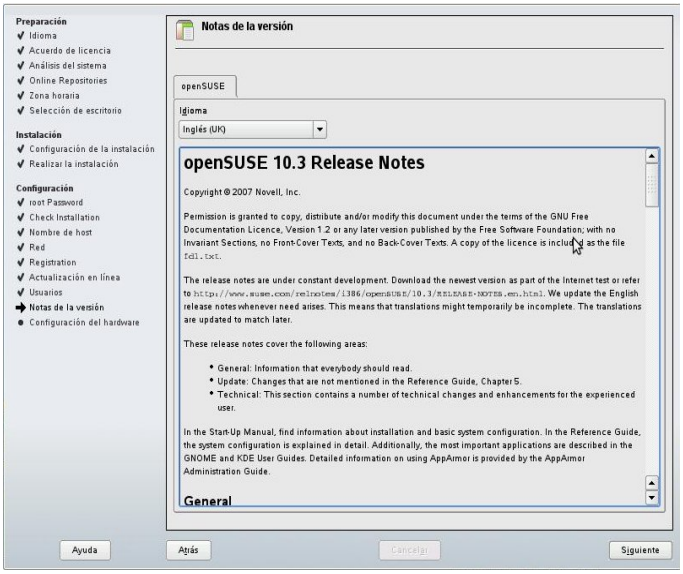

Fig. 66 Notas de versión

¡Configuración de Hardware, el último paso para finalizar! solo nos dice si la configuración que esta es la adecuada para nuestro equipo, si no lo es lo cambiamos, en caso de serlo o no sabemos, lo podemos cambiar una vez que ya tenemos todo instalado, así que no es un paso de suma importancia tener bien configurado nuestro hardware en este paso, le damos siguiente.

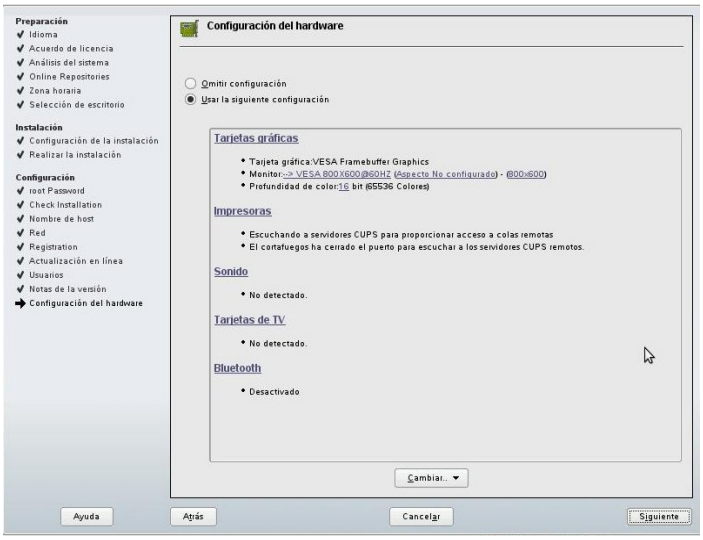

Fig. 67 Configuración de hardware

Ya tenemos un Sistema operativo GNU/Linux instalado y listo para ser usado, Bienvenido a OpenSuse

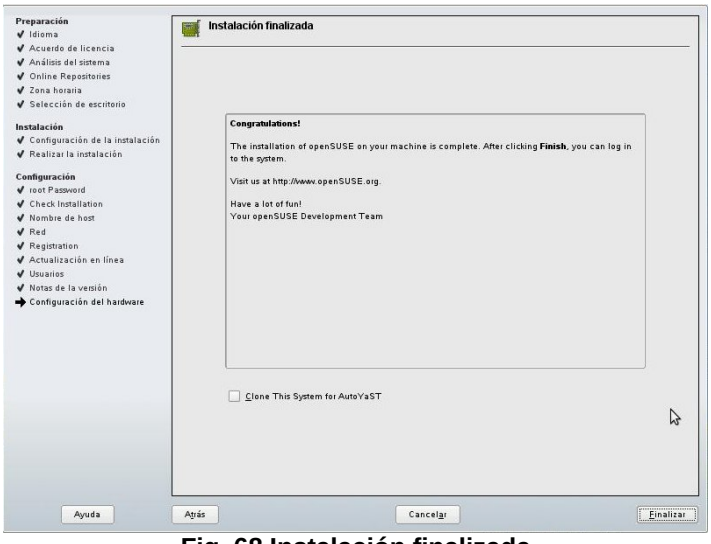

Fig. 68 Instalación finalizada

Listo ya tenemos nuestro sistema operativo OpenSuse 10,3 y a continuación mostramos el Grub

Fig. 69 Grub de OpenSUSE

16.3 Instalación de MYSQL

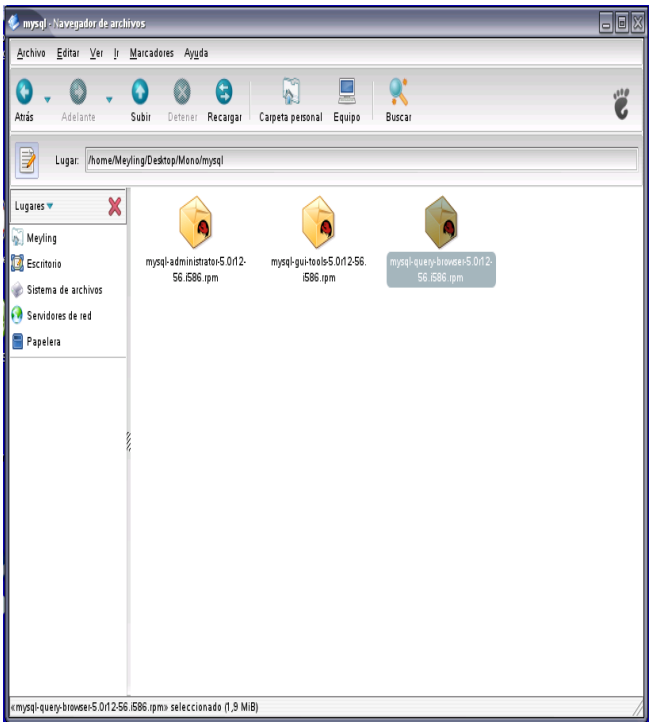

Fig. 70 Lista de paquetes a instalar

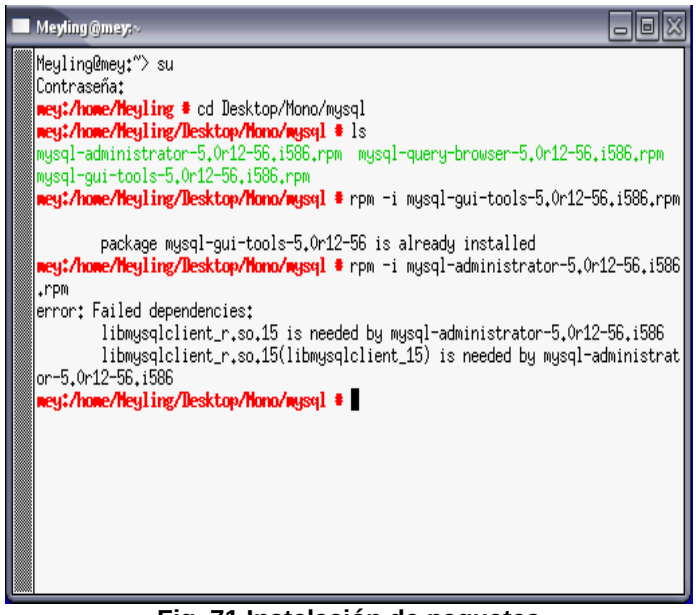

Fig. 71 Instalación de paquetes

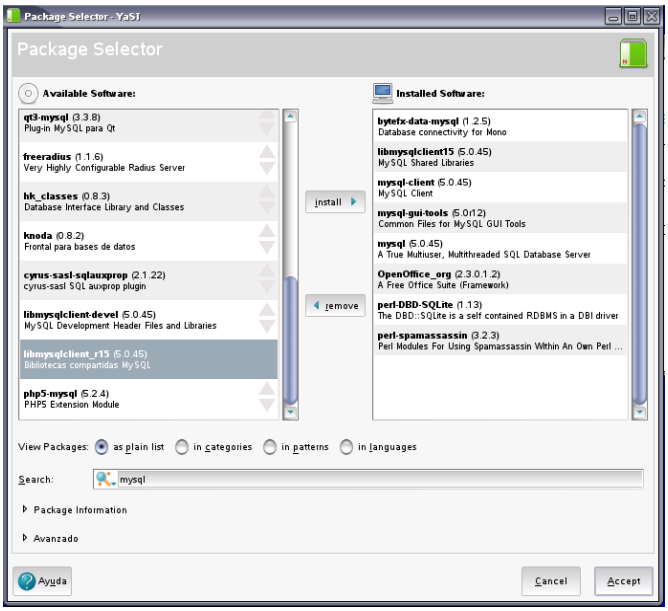

Fig. 72 Utilización del yast para instalar el GDBD MySQL

16.4 Instalación del Driver MySQL para una aplicación en monoDevelop

Para realizar la conexión a MySQL se uso el driver de MySQL MySql Connector/.Net que podemos descargar desde la página de MySQL, la cual se descargo del archivo "mysql-connector-net-version-noinstall.zip", el cual contiene la librería ensamblada del conector.

Una vez descargado el archivo es necesario instalar la librería en el GAC (Global Assembly Cache) para que sea accesible desde el Framework, en este caso descomprimimos "mysql-connector-net-1.0.8-RC-noinstall.zip" y copiamos el directorio y su contenido en "/home/Projects/mysql-connector-net".

Para instalarlo en el GAC solo fue necesario hacer lo siguiente (como root):

~> su

$ cd /home/Projects/mysql-connector-net

$ gacutil -i MySql.Data.dll

MySql.Data installed into the gac (/usr/lib/mono/gac)

Ahora es necesario crear la base de datos en MySQL y un usuario asociado a su acceso. Se creo una base de datos llamada "bd_gimnasio". (Utilizada en la aplicación principal de este trabajo monográfico).

Agregamos un nuevo usuario llamado "root" con contraseña "mono" que tenga acceso total a la base de datos "bd_gimnasio".

Ahora es momento de crear nuestro nuevo proyecto Gtk#, como siempre desde el menú "Archivo" en la opción "Nuevo Proyecto...".

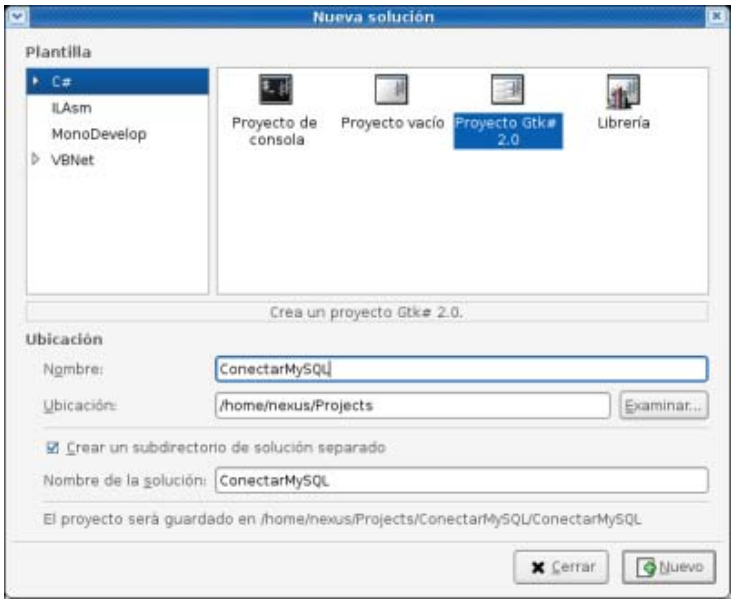

Fig. 73 Nuevo proyecto

Lo primero que debemos hacer es agregar las referencias apropiadas para el uso del MySQL Connector/.NET, y se logra dando un click derecho sobre la línea de "Referencias" en la ventana de "Solución" y pulsamos sobre "Editar Referencias".

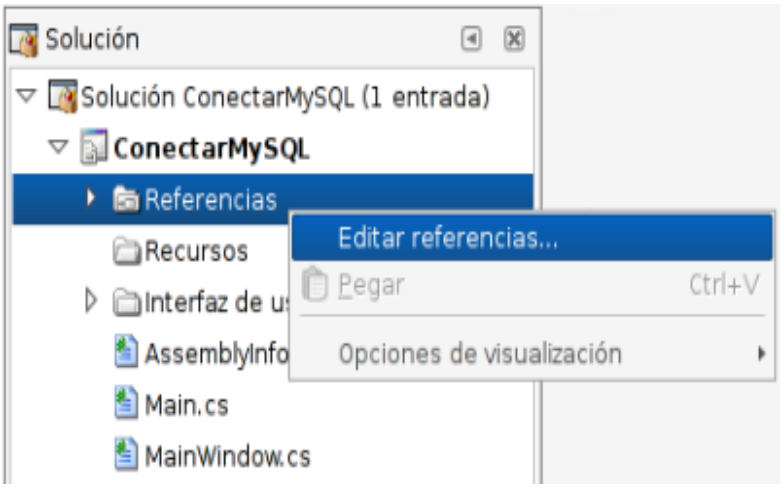

Fig. 74 Editar Referencia

Después vamos a la pestaña "Paquetes", donde debemos encontrar y activar la referencia "System.Data".

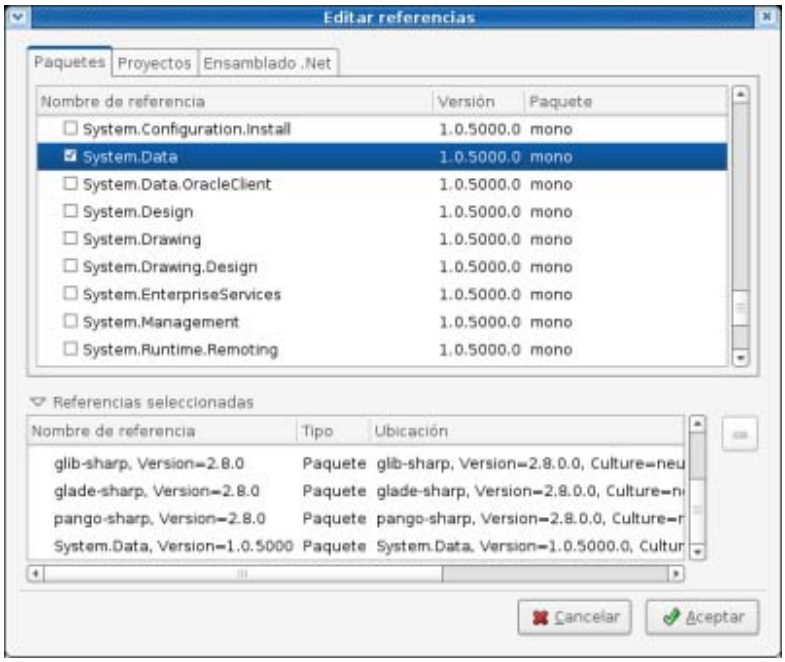

Fig. 75 Soporte de Base de Datos

Por último vamos a la pestaña "Ensamblado .Net", donde seleccionamos el archivo "MySql.Data.dll" y pulsamos sobre el botón "Añadir". Esto es así debido a que MonoDevelop no soporta los ensambles referenciados por GAC como paquetes.

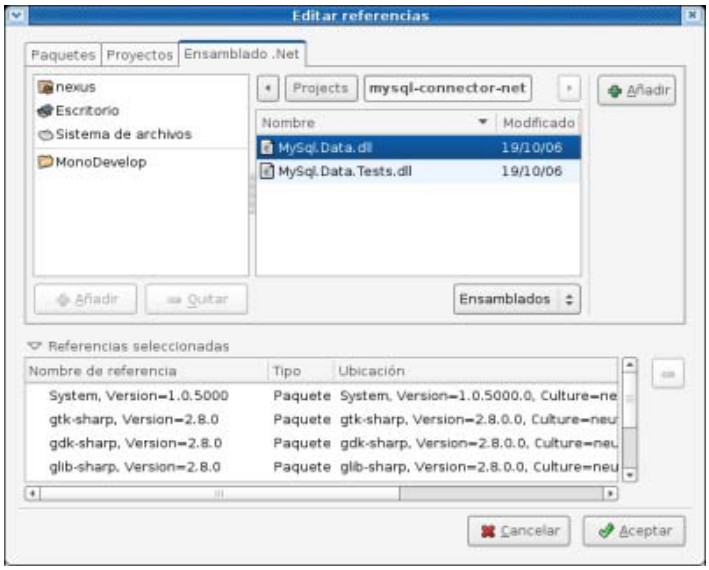

Fig. 76 Agregar Controlador o Driver

Ahora solo nos resta escribir el código que realizará la conexión y consulta sobre la Base de Datos, la cual quedará de la siguiente manera:(ver código de la clase Conexión de este trabajo monográfico)

```
//cadena de conexión a la base de datos
string connectionString =
"Server=localhost;" +
"Database=bd_gimnacio;" +
"User ID=root;" +
"Password=mono;" +
"Pooling=false";
 //consulta para extraer la información de la tabla
string myQuery = "SELECT id, nombre, apellidos FROM usuarios;";
// Se crea el objeto de conexión
Connection myConnection = new MySqlConnection(connectionString);

// Abre la conexión
                myConnection.Open();
                // Crea el objeto de ejecución del query
        MySqlCommand myCommand = new MySqlCommand(myQuery, myConnection);
     //cierra la conexión
myConnection.Close();
```

16.5 Descripción General de Monodevelop

Realizaremos una breve descripción de nuestro creador de interfaces gráficas, creando dos proyectos sencillos para demostrar sus facilidades.

Creación del proyecto Hola mundo indicando paso a paso como empezar a diseñar un proyecto.

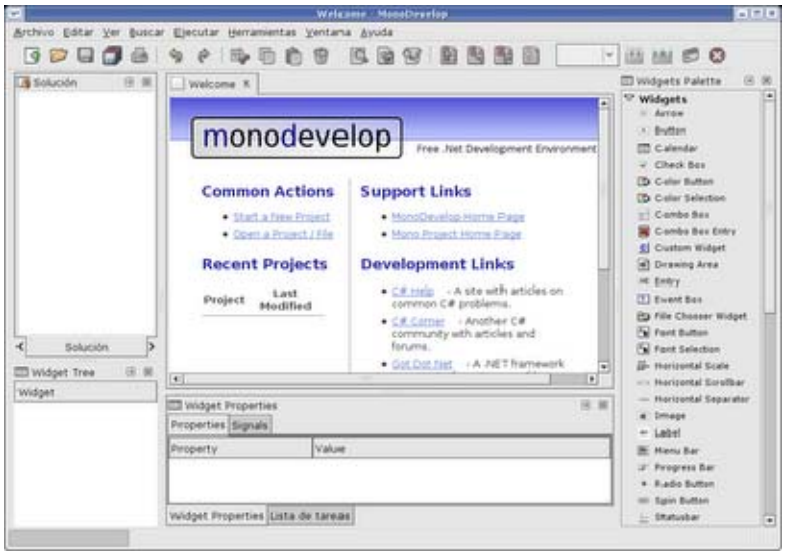

Fig. 77 Ventana de inicio del creador de interfaces

Lo primero que hay que hacer es crear una nueva solución, la cual se puede crear de forma automatizada desde el menú "Archivo" y "Nuevo Proyecto...", la cuál nos abrirá una ventana llamada "Nueva Solución", desde ahí es necesario seleccionar un "Proyecto Gtk# 2.0" y ponerle un nombre a la solución.

Por ejemplo "Prueba", por último pulsamos sobre "New" para que Monodevelop se encargue de crear todos los archivos necesarios para comenzar a trabajar.

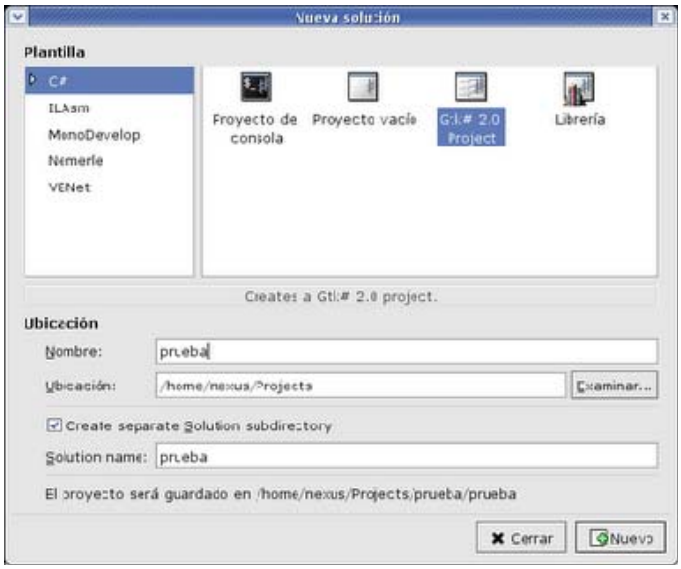

Fig. 78 Nueva solución

De entre todos los archivos de la solución, existen dos que son importantes "Main.cs" y "MainWindow.cs" los cuales se pueden ver en la pestaña de "Solución"

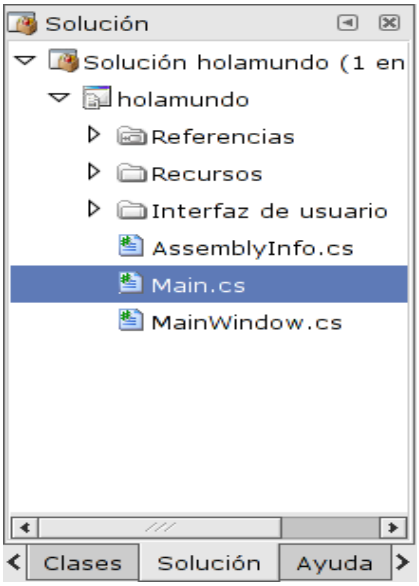

Fig. 79 Solución

Ejemplo 1: Hola Mundo: Para empezar a trabajar con el diseño de la nueva aplicación se debe seleccionar "MainWindow.cs" y pulsar sobre el botón "Diseñador" en la parte inferior de esa ventana. Con lo anterior lograremos ver la venta que podemos usar para el diseño.

Diseño Arquitectónico “Holamundo”

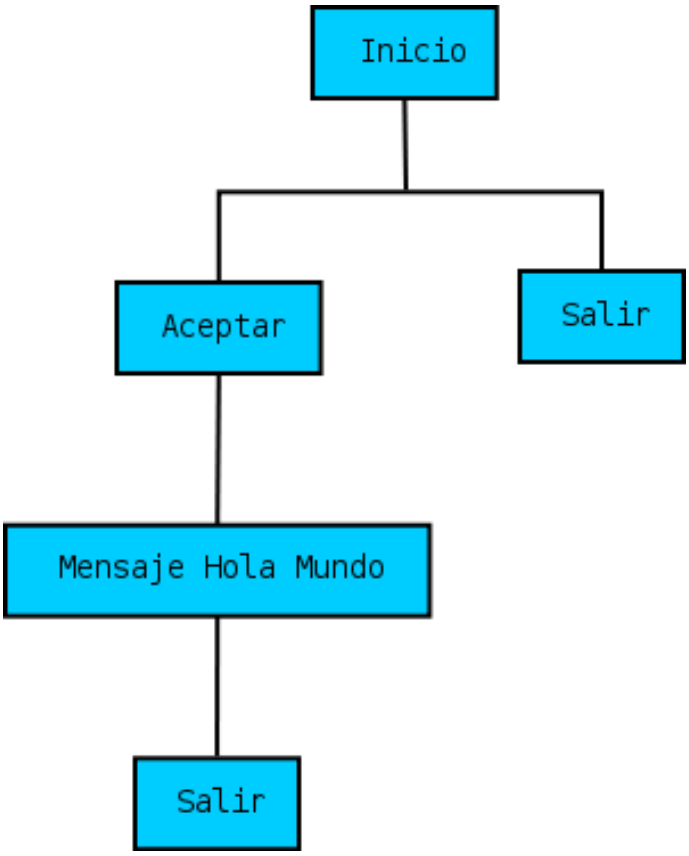

Fig. 80 Diseño Arquitectónico “Holamundo”

Diseño de Clase “Holamundo”

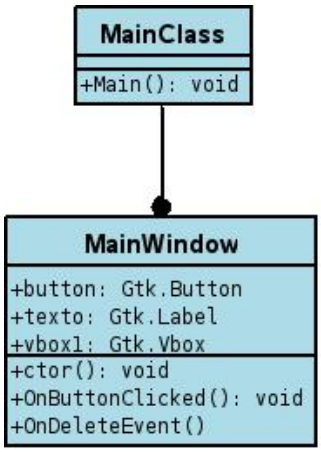

Fig. 81 Diseño de Clase “Holamundo”

Fig. 82 Diseñador

Es importante aclarar que Stetic no trabaja como el diseñador de formularios de MS Visual Studio, ya que este no se basa en coordenadas para la construcción y dibujo de los controles, más bien es necesario ocupar los Widgets (componentes) "HBox" y "VBox" e imaginarnos como quedaría el formulario usando tablas.

Fig. 83 Contenedores

A lo que nos referimos es que los controles o componentes deben estar dentro de alguno de estos u otros contenedores

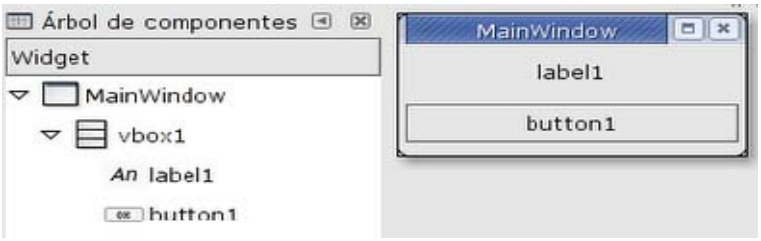

Fig. 84 Controles o componentes

Ahora es necesario ponerle nombre a los controles, esto se logra usando la pestaña de "Propiedades de Componente", es tan simple como seleccionar el control y modificar sus propiedades de forma muy fácil.

En este proyecto uso los siguientes valores:

Para el "MainWindow"

Window Title: Proyecto Hola Mundo

Window Position: Centered

Para el "label"

Widget name: texto

Label: Presiona el botón

Justification: Center

Para el "button"

Widget name: boton

Buton Type: Text and Icon

Icon: gtk-yes

Label: Aceptar

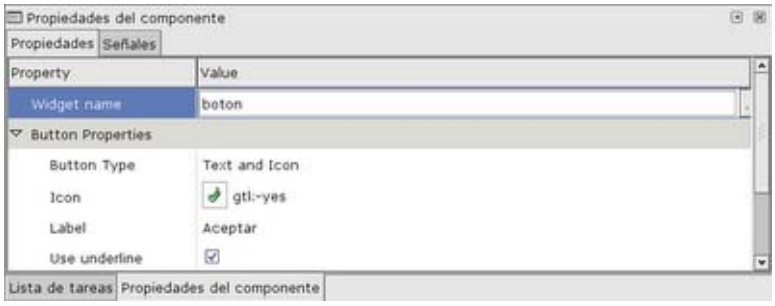

Fig. 85 Propiedades del componente

El siguiente paso a seguir es declarar la acción del botón, que cuando se de un clic este haga algo, para realizar esta declaración es necesario seleccionar el botón y después ir a la pestaña "señales" en las "Propiedades del componente". Ahí podremos ver un listado de comportamientos de control, buscamos el que se usa al pulsar el botón que se llama "Clicked". Al lado de este evento veremos un mensaje que dice "Click here to add a new handler", así que le hago caso y le doy un click para ponerle un nombre... se me ocurre por ejemplo: "boton_Clicked", o sea "boton" ya que es el nombre del control y agregarle el sufijo del nombre del evento, para que nuestro código tenga coherencia.

Fig. 86 Señales del componente

Ya casi terminamos de usar el diseñador para empezar a programar, lo único que hace falta es seleccionar el "label" y pulsar sobre el botón "Asociar a Campo", y hay que hacer exactamente lo mismo con el "button", esto sirve para que **Monodevelop** declare el uso de los controles en la clase.

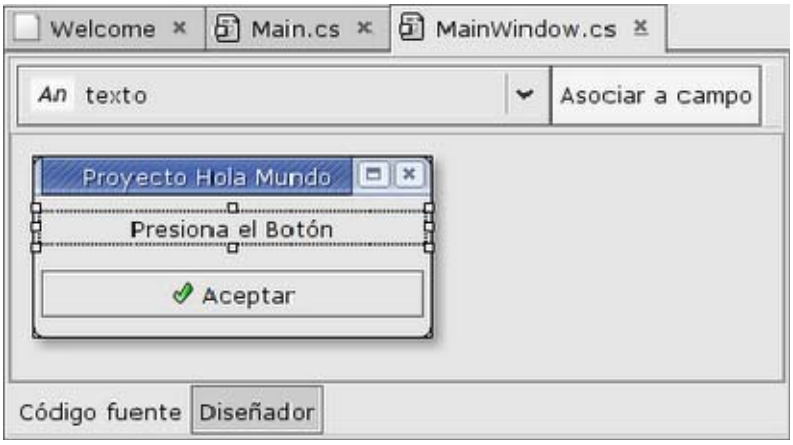

Fig. 87 Diseñando el botón

Listo, ahora es tiempo de ir a trabajar sobre el código pulsando sobre el botón "Código Fuente" del "MainWindow.cs" y veremos lo siguiente:

```
using System;
usingGtk;
public class MainWindow: Gtk.Window
{
protected Gtk.Label texto;
protected Gtk.Button boton;
public MainWindow (): base ("")
{
Stetic.Gui.Build (this, typeof(MainWindow));
}
 protected void OnDeleteEvent (object sender, DeleteEventArgs a)
{
Application.Quit ();
a.RetVal = true;
}
protected virtual void boton_Clicked(object sender, System.EventArgs e)
{
}
}
```

Veremos varias diferencias entre la versión de cuando iniciamos el proyecto a este momento. Las más significativas son:

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

Estas líneas fueron agregadas por MonoDevelop al momento de nombrar el evento "Clicked".

```
protected Gtk.Labeltexto;
protected Gtk.Button boton;
```

Estas líneas fueron agregadas cuando pulse sobre el botón "Asociar a Campo" seleccionado el "label" y el "button".

Ahora si, es el momento de escribir código!!!

```
protected virtual void boton_Clicked(object sender, System.EventArgs e)
{
texto.Text = "Hola Mundo!!!";
}
```

Ya quedó, eso fue todo, únicamente una sola línea de código (agregar la tercera línea) y la aplicación ya esta lista para ser compilada, así de fácil, ¿no es impresionante?. Podemos probar como funciona nuestra aplicación de forma rápida, pulsando sobre el icono de ejecutar.

Fig. 88 Ejecutar

Y veremos nuestra aplicación funcionando correctamente!!!

Fig. 89 Proyecto holamundo ejecutado

Ejemplo 2: Visor de imágenes

El proyecto cargará una imagen al ejecutarlo, si se da clic sobre el formulario nos aparecerá una ventana donde se encuentran las archivos con nuestras imágenes, seleccionamos una y el formulario se nos actualiza con imagen seleccionada.

Diseño Arquitectónico "Visor de Imágenes"

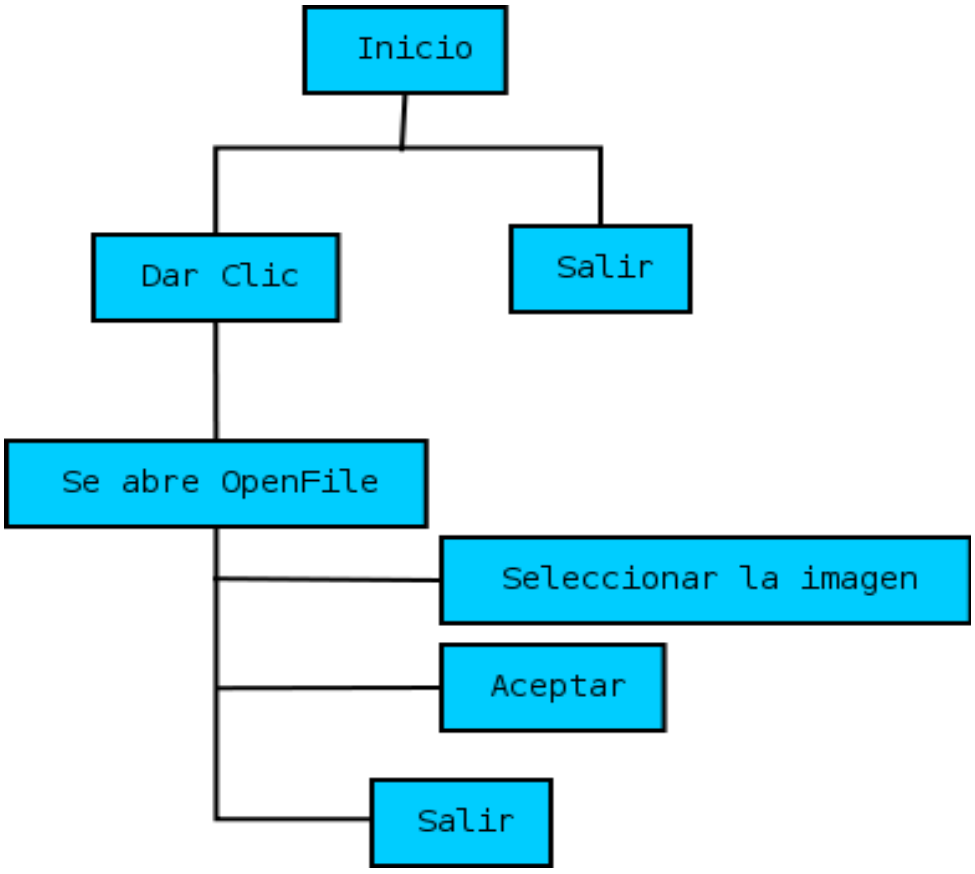

Fig. 90 Diseño Arquitectónico "Visor de Imágenes"

Diseño de Clase “Visor de Imágenes”

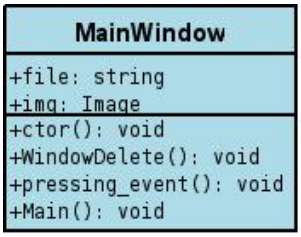

Fig. 91 Diagrama de clase de “Visor de Imágenes”

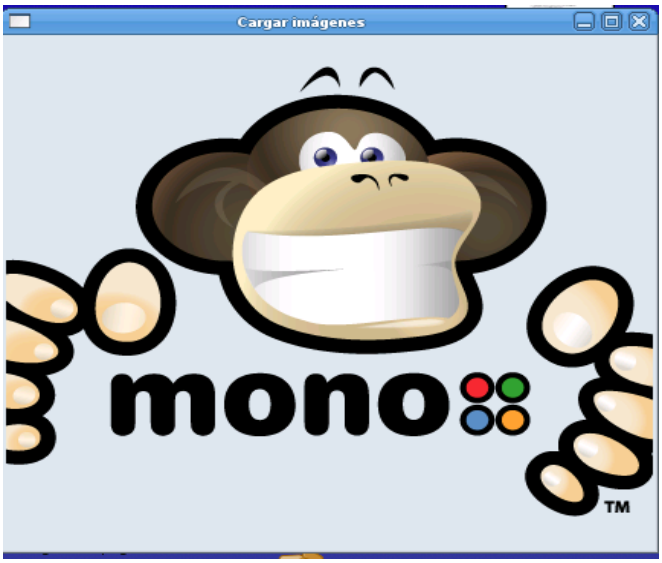

Fig. 92 Proyecto ejecutado

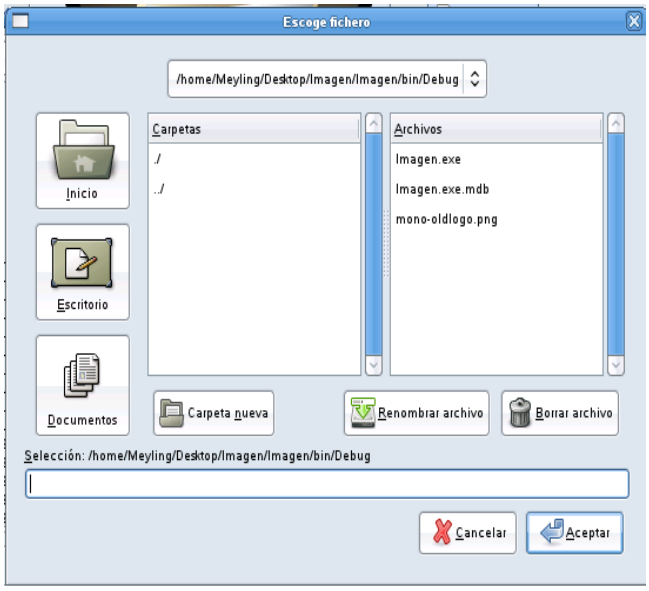

Fig. 93 Ventana de búsqueda de imagen nueva

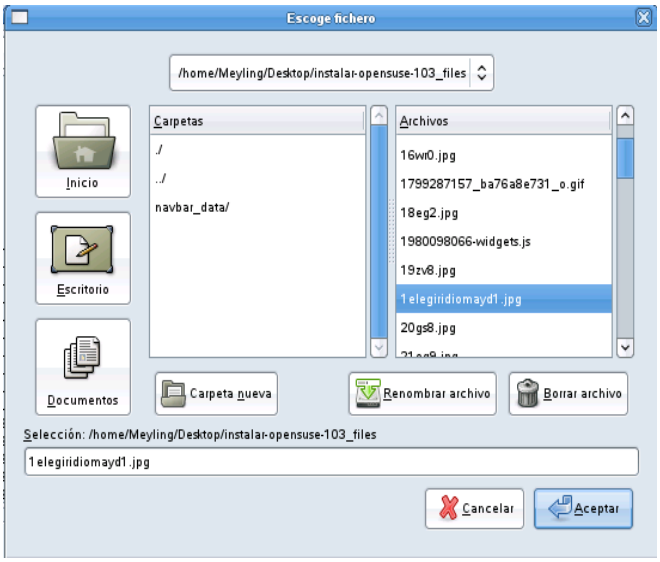

Fig. 94 Selección de la nueva imagen

Fig. 95 Formulario actualizado con nueva imagen

Visor.cs

```
using System;
using Gtk;
public partial class MainWindow: Gtk.Window
{
public MainWindow (): base (Gtk.WindowType.Toplevel)
{
        Build ();
}
protected void OnDeleteEvent (object sender, DeleteEventArgs a)
{
        Application.Quit ();
        a.RetVal = true;
}
}//fin del main
using System;
using Gtk;
using System.IO;
using Gtk;
namespace Imagen
{
        public class Imagen {
         string file;
        Image img;
```

```
public static void Main (string[] args) {
 new Imagen (args);
 }
 public Imagen (string[] args)
{
        Application.Init ();
        Window win = new Window (WindowType.Toplevel);
        win.Title = "Cargar imágenes";
        win.SetDefaultSize(300,300);
        win.DeleteEvent += new DeleteEventHandler (WindowDelete);
         // Añadir caja de eventos
        EventBox;
        eventbox = new EventBox ();
        win.Add (eventbox);
        eventbox.Show();
         // añadir imagen
        img = new Image("mono-oldlogo.png");
        eventbox.Add(img);
        img.Show();
        img.SetSizeRequest(100,100);
```

```
 // capturamos el evento
eventbox.ButtonPressEvent+= new ButtonPressEventHandler (pressimg_event);
                    eventbox.Realize();
                     win.ShowAll ();
                    Application.Run ();
}
void pressimg_event (object obj, ButtonPressEventArgs args)
{
          FileSelection fs = new FileSelection ("Escoge fichero");
          fs.Run ();
          file = fs.Filename;
          fs.Hide ();
           img.File = file;

}
static void WindowDelete(object obj, DeleteEventArgs args)
          {
          Application.Quit();
          }
    }
}
```