Desarrollo de Software con Mono, una Implementación Libre de .NET, Multiplataforma e Independiente del Lenguaje

Larrateguy, Luis Ignacio ‡ *luisignacio@larrateguy.com.ar*

Pividori, Milton Damián ‡ *miltondp@gmail.com*

Sandrigo, César Matías † *cmsandrigo@gmail.com*

‡ CIDISI - FRSF UTN

† FRSF UTN

Resumen

Se presenta un desarrollo como prueba de concepto en la integración de tecnologías, utilizando Software Libre y multiplataforma. Se utilizó la plataforma de desarrollo Mono, implementación libre de .NET. Se probaron tecnologías como Remoting, Web Services, independencia del lenguaje de programación y toolkits gráficos libres. Además se utilizó el entorno de desarrollo integrado libre MonoDevelop. Se abordó la solución con técnicas orientadas a objetos y se aplicaron patrones de diseño. Uno de los objetivos propuestos por los autores, fue aprender a utilizar tecnologías libres y modernas utilizando Software Libre. En el caso de estudio se probó portabilidad entre Ubuntu GNU/Linux y Microsoft Windows.

Introducción

Ventajas del Software Libre.

- Para **clientes**: la independencia sobre las empresas proveedoras de software, habilitándolos a contratar a los profesionales que prefiera para realizar tareas sobre su sistema.
- Para **usuarios:** la seguridad de estar trabajando con estándares abiertos, y la posibilidad de elegir entre varias alternativas.
- Para **desarrolladores:** beneficio de los aportes que puedan realizar sus pares, de contar con una gran comunidad para el soporte, incluso de realizar trabajo diferencial sobre una pieza de software existente.

Característica de .NET: independencia del lenguaje.

- Se utilizaron varios lenguajes en el desarrollo, como C#, Java y Boo, y comunicándolos entre ellos a nivel de clase.
- Se trabajó con un ***lenguaje común*** entre los individuos: la ***Orientación a Objetos.***

¿Qué es Mono?

"Mono es una plataforma de software diseñada para permitir a los desarrolladores crear fácilmente aplicaciones multiplataforma. Es una implementación *open source* del Framework .NET de Microsoft basado en estándares ECMA para C# y el *Common Language Runtime*".

¿Qué es GTK#?

Gtk# es el nombre de los *bindings* correspondientes de Gtk+ para la plataforma Mono/.NET. Es posible utilizar este toolkit gráfico desde cualquier lenguaje soportado por la plataforma, no sólo C#.

Remoting

- Conjunto de servicios que permite la comunicación entre procesos.
- Los procesos pueden residir en la misma computadora, o en diferentes estando conectadas por una LAN o Internet.
- Permite la comunicación entre objetos de diferentes dominios de aplicación o procesos.
- Es posible utilizar diferentes protocolos de transporte, optar por varias formas de serialización, configurar el esquema de tiempo de vida de los objetos y elegir entre varios modos de creación de los mismos.
- Se puede ver un gráfico de la arquitectura de Remoting en la ***Ilustración 1***.

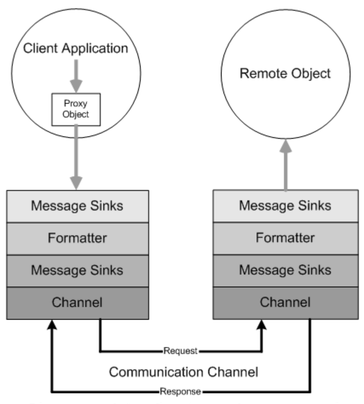

Ilustración 1: Arquitectura de Remoting

Web Services

- Son interfaces accesibles a la funcionalidad de una aplicación, construidas sobre estándares de tecnologías que utilizan Internet.
- Se basan en estándares que definen la comunicación entre el proveedor, consumidor y publicador de servicios.

Planteo de un Problema a Resolver

Crear un programa que permita enviar "mensajes instantáneos" a través de una red. Las condiciones del caso de uso son las siguientes:

- Clientes que se pudieran comunicar enviando mensajes de texto plano.
- Ver contactos conectados.
- Poder enviar y recibir mensajes mediante un sistema web.

Para resolver este problema se plantearon las siguientes restricciones:

- Los clientes usarían Remoting para enviarse los mensajes.
- Programar en distintos lenguajes de programación.
- El cliente web debía funcionar utilizando Web Services.
- Utilizar exclusivamente Software Libre.

Arquitectura

- Arquitectura ***Cliente-Servidor*** para los desarrollos sobre Mono/.NET.
- Arquitectura distinta para los clientes que utilizarán Web Services. Cada una de las acciones son expuestas en forma de servicios (conectarse, enviar mensajes, recibir mensajes, ver lista de contactos, desconectarse), los cuales son ejecutados por ***"clientes virtuales"***, que representan al cliente web en su ausencia.

La ***Ilustración 2*** muestra la arquitectura propuesta para resolver el problema de interoperabilidad y poder hacer una prueba de integración, no sólo de clientes que corran sobre la plataforma .NET, sino con clientes implementados en otra plataforma (como PHP) utilizando Web Services.

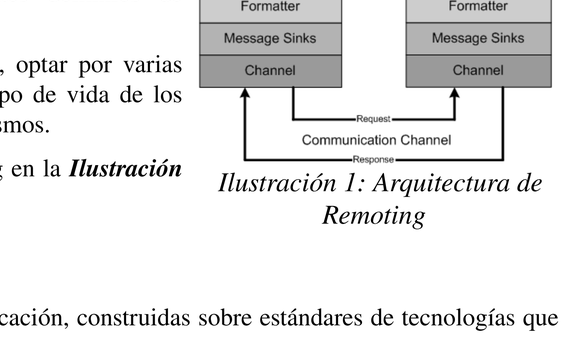

Ilustración 2: Arquitectura propuesta

Implementación y Despliegue

En la ***Ilustración 3***, se puede observar como fue implementada esa arquitectura utilizando Remoting.

Especializando *MarshalByRefObject*, se obtienen clases que pueden generar instancias únicas tanto del servidor como de los clientes. El Servidor mediante un stub de *ClienteRemoto* puede informar de los eventos a cada cliente, y éstos pueden enviar un mensaje al objeto remoto *ControladorConexiones* mediante un stub creado por Remoting. En la ***Ilustración 4***, se muestran los paquetes que se implementaron por separado, primero fijando las interfases necesarias para poder continuar el desarrollo en forma conjunta y distribuida.

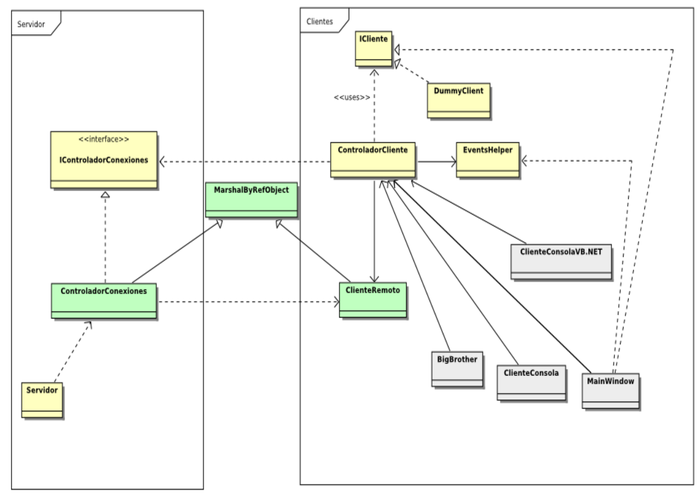

Ilustración 3: Diagrama de clases correspondientes al cliente y al servidor

Implementación en Distintos Lenguajes

Cada paquete está programado en un lenguaje diferente. Se utilizó **C#**, **Boo**, **Java** (mediante IKVM.NET) y **VB.NET**. Con otros lenguajes también se hicieron pruebas pero no estaban lo suficientemente maduros.

GtkGUI implementa una interfaz simple pero completa de mensajería instantánea. La interfaz gráfica en sí, está diseñada utilizando ***Glade***. Glade es un diseñador gráfico de interfaces. El resultado es guardado en un archivo XML, permitiendo que el mismo diseño de interfaz se utilice en otra plataforma o lenguaje de programación, siempre y cuando existan enlaces a bibliotecas Gtk+/Glade.

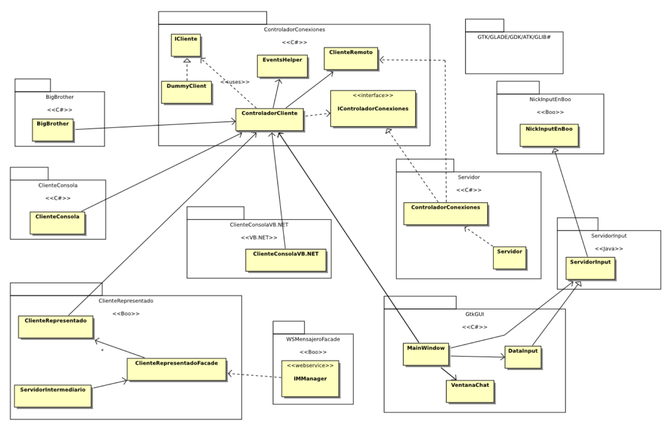

Ilustración 4: Diagrama con todas las clases del sistema

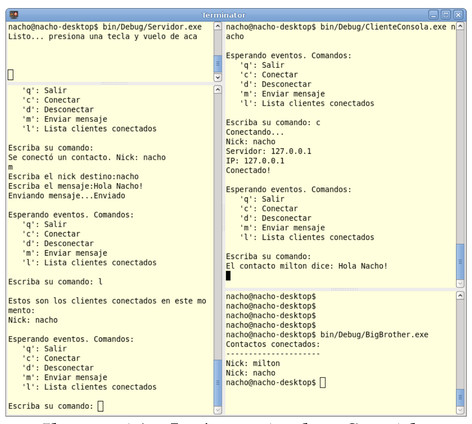

Ilustración 5: 4 terminales. Servidor, ClienteConsola C#, ClienteConsolaVB.NET, y BigBrother mostrando los clientes conectados

Para conectarse cada cliente debe elegir un nickname, indicar la IP del servidor y el puerto donde éste está escuchando, como se puede apreciar en la ***Ilustración 6.***

En la ***Ilustración 7*** se puede apreciar una ventana de chat y la lista de contactos corriendo en Windows Vista y Ubuntu respectivamente.

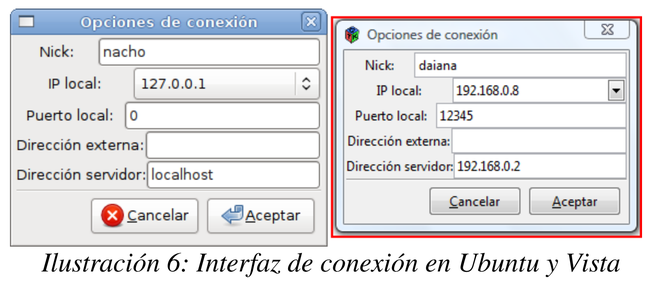

Ilustración 6: Interfaz de conexión en Ubuntu y Vista

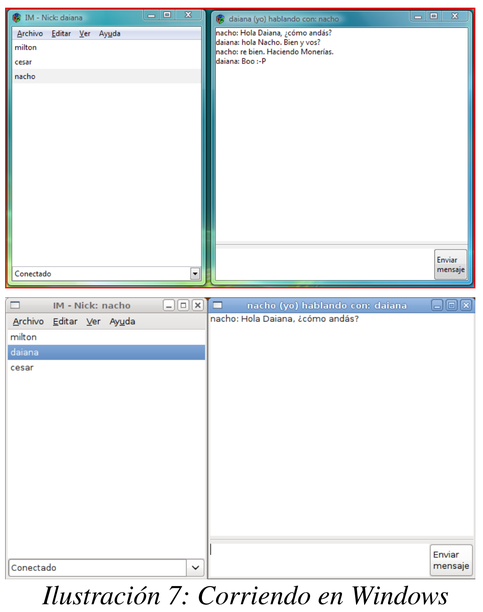

Ilustración 7: Corriendo en Windows Vista, con Framework .NET y GTK#.; y en Ubuntu con Mono.

Implementación del Cliente Web Mediante Web Services

El cliente web de la ***Ilustración 8*** consiste de dos partes. Una parte "cliente" que es la que utiliza el usuario final, y otra "servidor" que es la que se comunica con el Web Service y consume estos servicios. La parte del servidor fue implementada con PHP, utilizando la biblioteca NuSOAP que es una API para acceder a los servicios brindado por un Web Service. La parte del cliente fue implementada, utilizando llamadas asíncronas con Javascript de fondo, con el fin de chequear por eventos en el servidor. Estas llamadas se repetían en un modo de "polling" con un intervalo de tiempo determinado.

El usuario del cliente web, ve una sola ventana de chat, en donde se puede comunicar individualmente con cada persona, pero recibe los mensajes en una misma pantalla.

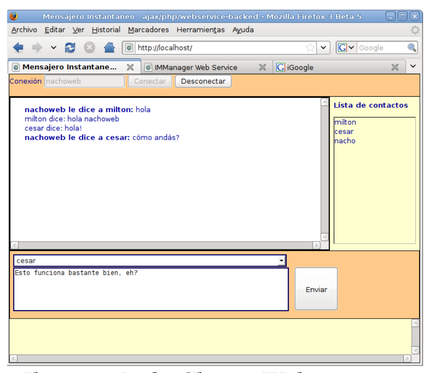

Ilustración 8: Cliente Web escrito en PHP, corriendo sobre un servidor Apache

Conclusión

Las tecnologías abiertas y particularmente el Software Libre han ido avanzando y ganando terreno en la comunidad y el mundo empresarial. Cada día se ve más necesidad de personal capacitado en esta área. El desarrollo del presente trabajo permitió a los autores conocer una nueva plataforma que se integra con otras existentes, privativas y libres. El involucrarse con estas tecnologías y seguir de cerca su evolución permite ir evaluando la madurez de los productos y la aceptación que va teniendo en la industria.

La arquitectura resultante de la *prueba de concepto* permitió probar funcionalidades de integración de tecnologías heterogéneas y ganar experiencia para futuros desarrollos utilizando Software Libre.