Usando GNOME
para el desarrollo rápido de aplicaciones

RESUMEN

El desarrollo de aplicaciones al interior de las empresas es una actividad altamente difundida a nivel nacional, el uso de tecnologías libres se ha convertido de manera creciente en un anhelo para los desarrolladores y profesionales de las TI, sin embargo aún existe diversas barreras de entrada que evitan el uso de este tipo de tecnologías al interior de las empresas, entre ellas podemos mencionar; El bajo nivel de conocimiento en los equipos de desarrollo, donde es común encontrar prácticas obsoletas o erróneas, además del poco dominio de técnicas y principios de la ingeniería del software tales como el control de versiones o programación orientada a objetos. La gran cantidad de opciones y herramientas disponibles en el software de código abierto y las tecnologías libres es otra de las barreras que a menudo los equipos de desarrollo deben sortear, sin embargo cada vez es más notoria la elección de herramientas multiplataforma, las que facilitan los procesos de migración al interior de la organización.

El proyecto GNOME [1] tiene como uno de sus objetivos el entregar una completa plataforma de desarrollo para los ambientes UNIX y Linux, a su vez muchas de sus herramientas y aplicaciones están portadas a otras plataformas tales como Win32 y MacOS X, esto convierte a la plataforma de desarrollo en una opción muy atractiva para realizar proyectos que puedan funcionar indistintamente en varias plataformas y sistemas operativos. Sin embargo todo el desarrollo base del proyecto así como librerías y herramientas se encuentra escrito en C, lenguaje que resulta muy árido para la mayoría de los desarrolladores de aplicaciones de gestión, dada las características de bajo nivel propias del lenguaje. Sin embargo actualmente existen una variedad de opciones para usar las librerías de GNOME en otros lenguajes de alto nivel tales como Java, C# y Python.

El presente artículo muestra las diversas alternativas y herramientas de desarrollo usando GNOME que pueden ser encontradas hoy en día, entre ellas cuentan entornos de desarrollo integrados (IDE), lenguajes de desarrollo, diseñadores de interfaces y documentación relacionada con GNOME

Palabras Claves: GNOME, Desarrollo de Aplicaciones, Lenguajes de Alto Nivel.

Lenguajes para desarrollar usando GNOME.

Las librerías bases del proyecto GNOME se encuentran escritas en C al igual que la mayoría del software desarrollado en ambientes UNIX, sin embargo dentro del proyecto existe el concepto de envolturas (bindings), el cual permite que una librería de desarrollo pueda ser utilizada en diversos lenguajes. La principal librería de desarrollo de GNOME es GTK+ [2], en ella se encuentran la mayoría de los widgets (objetos gráficos) utilizados por las aplicaciones de GNOME, además esta librería esta portada a otras plataformas tales como Win32 y MacOS X.

Existe diversos lenguajes de alto nivel y multiplataforma soportados por las envolturas de GTK+ sin embargo en este artículo se centrará sobre tres lenguajes de programación dado su alto grado de utilización, facilidad de aprendizaje y aceptación por la comunidad:

- Java [3]: es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales. Consta del lenguaje de programación, la máquina virtual o JRE (Java Runtime Environment) y la biblioteca estándar del lenguaje.
- C# [4]: es un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado, como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes. C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic. Por otra parte como se demuestra más adelante la plataforma .NET no solamente cuenta con C# también existen otros populares lenguajes tales como Visual Basic.
- Python [5]: es un lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas. Python es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

Java para GNOME

El proyecto Java-GNOME [6] es un conjunto de envolturas para GNOME y GTK+ que permiten crear aplicaciones usando java. El proyecto es parte oficial del conjunto de envolturas para GNOME [7].

Instalación

Para los ejemplos de instalación se supone contar con una máquina instalada con Ubuntu Linux 5.04 [8] y conectada a Internet, para instalar los paquetes java-gnome2 se debe ejecutar lo siguiente

```
$ sudo aptitude install libgtk2-java libgnome2-java libglade2-java libgconf2-java
```

Un ejemplo HelloWorldGtk.java:

```
import org.gnu.gtk.Button;
import org.gnu.gtk.Gtk;
import org.gnu.gtk.Window;
import org.gnu.gtk.WindowType;
import org.gnu.gtk.event.ButtonEvent;
import org.gnu.gtk.event.ButtonListener;
import org.gnu.gtk.event.LifeCycleEvent;
import org.gnu.gtk.event.LifeCycleListener;

public class HelloWorldGTK {

     public HelloWorldGTK() {

          Window window;
          Button btn;

          window     = new Window(WindowType.TOPLEVEL);
          btn   = new Button("Hello!");

          //listen for click events on the button...
          btn.addListener(new ButtonListener() {
               public void buttonEvent(ButtonEvent e) {
                    if (e.isOfType(ButtonEvent.Type.CLICK)) {
                         System.out.println("Hello cruel world!");
                    }
               }
          });
          window.addListener(new LifeCycleListener() {
               public void lifeCycleEvent(LifeCycleEvent arg0) {}

               //quit the app on close event...
               public boolean lifeCycleQuery(LifeCycleEvent arg0) {
                    System.out.println("Finishing app...");
                    Gtk.mainQuit();
                    return false;
               }
          });

          window.add(btn);
          window.setDefaultSize(150, 50);
          window.showAll();
     }
     public static void main(String[] args) {

          Gtk.init(args);
          new HelloWorldGTK();
          Gtk.main();
     }
}
```

La compilación se realiza utilizando las librería GTK+:

```
$ javac -classpath /usr/share/java/gtk2.6.jar HelloWorldGTK.java
```

Para ejecutar el ejemplo:

```
$ java HelloWorldGTK.java
```

El resultado en pantalla es el siguiente:

Figura 1 – Demo java-GNOME

Eclipse: entorno de desarrollo para java-gnome

El proyecto eclipse[9] es una comunidad de desarrollo de código abierto cuyo proyecto está enfocado a proveer un entorno y plataforma de desarrollo para la construcción de software. El proyecto fue originalmente lanzado por IBM pero más tarde fue liberado, actualmente cuenta con una solida comunidad de desarrollo, esta basado entre otros en GTK+ e incluso se distribuye como parte de Fedora Core 4 [10]. Uno de los elementos más atractivos de eclipse es su sistema de complementos (plugins), el cual permite extender el entorno de desarrollo con nuevas funcionalidades incorporando distintas tecnologías que facilitan el desarrollo rápido de aplicaciones.

Pese a que la cara más visible de eclipse es el entorno de desarrollo es justo mencionar que el proyecto es mucho más amplio, incluyen un conjunto importante de herramientas, tecnologías y conceptos que facilitan el desarrollo rápido de aplicaciones basadas en Java, las que se integran directamente al escritorio GNOME, los tres sub-proyectos son:

- Eclipse Proyect
- Eclipse Tools
- PDE (Plugin Development Enviroment)

Las herramientas integradas a eclipse operan en archivos del espacio de trabajo (workspace) del usuario. El espacio de trabajo consta de uno o más proyectos donde cada uno mapea a un directorio especificado por el usuario en el sistema de archivos. El usuario se comunica con eclipse por intermedio del marco de trabajo (workbench) que se inicia junto con eclipse. El marco de trabajo, es la interfaz de usuario de la plataforma. El mismo está compuesto de un conjunto de vistas y editores y perspectivas. Los editores permiten crear, modificar y guardar objetos, las vistas proveen información acerca de los objetos con los que se está trabajando en el marco de trabajo y las perspectivas proveen distintas formas de organización para el proyecto. Si bien se tiene varias perspectivas para el proyecto se puede usar una a la vez. Un ejemplo del aspecto de eclipse se puede observar en la Figura 2.

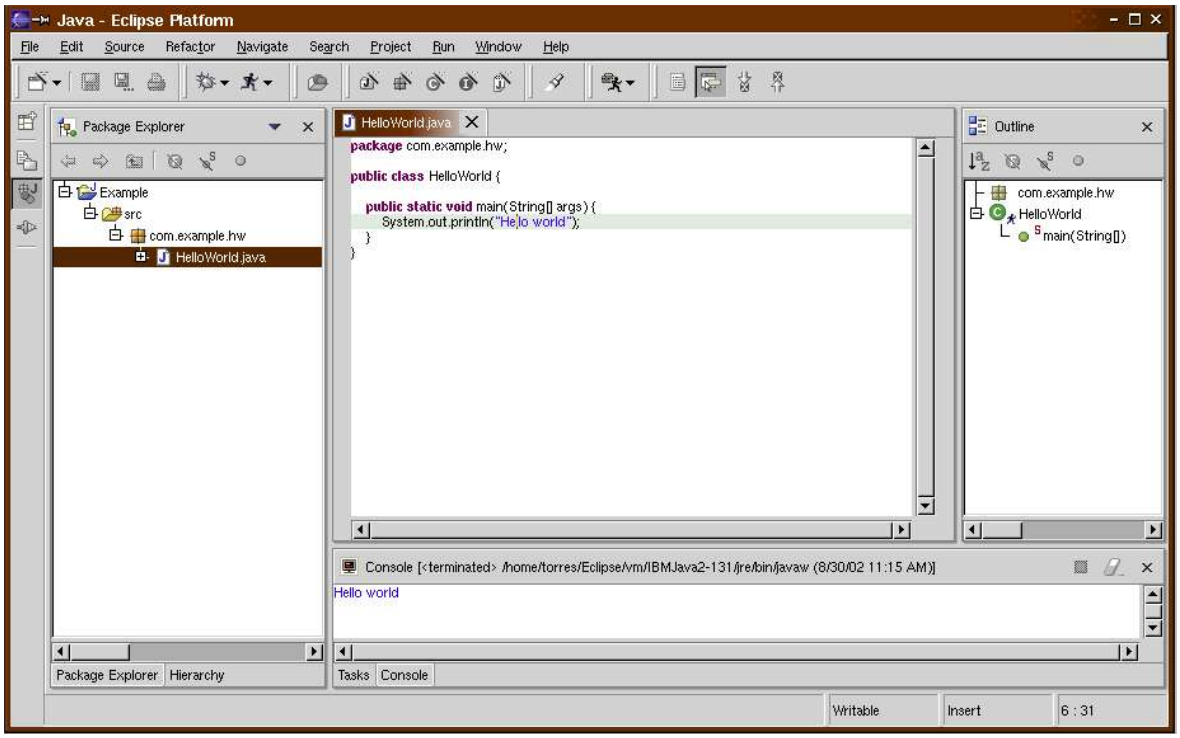

Figura 2 - Eclipse

.NET para GNOME

Los proyectos grandes de software deben de hacer frente a problemas que no existen en proyectos menores. Los programas que tienen un ciclo de vida largo deben hacer frente de distinta forma a la gestión de memoria que los programas pequeños. Llega un momento en el que los desarrolladores notan que han perdido demasiado tiempo escribiendo destructores, solucionando un problema de memoria, utilizando funciones no seguras de bajo nivel, y que han implementado demasiadas listas enlazadas. La plataforma .NET es una gran avance sobre todo en cuanto a productividad, de forma que aunque Microsoft haya desarrollado estas tecnologías teniendo en mente servicios WEB, el mayor beneficio de ésta es aumentar la productividad del programador. Este ha sido uno de los principales objetivos de Novell y su proyecto Mono [11], el cual busca dotar a la comunidad de software libre en general y a GNOME en particular de una potente herramienta de desarrollo que logre maximizar la productividad. Mono hace uso de las tecnologías de GNOME como parte de su implementación de la librería de clases, así es posible escribir aplicaciones en otras plataformas como Windows con la librería Gtk# [12], los cuales se integran sin problemas tanto en Linux como en otros sistemas operativos. Entre las características más destacables de Mono están:

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

- 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, PostgreSQL, OLE DB, en total hasta 27 bases de datos. Extensa librería de clases: Criptografía, HTTP, Bases de datos, GUI y muchas otras.

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

Un ejemplo en C# usando Gtk#:

```
using Gtk;
using System;
class Hello {
        static void Main() {
            Application.Init ();
            Button btn = new Button ("Hello!");
            btn.Clicked += new EventHandler (hello);
            Window window = new Window ("GTK# DEMO");
            window.SetDefaultSize(150, 50);
            window.DeleteEvent += delete_event;
            window.Add (btn);
            window.ShowAll ();
            Application.Run ();
        }
        static void delete_event (object obj, DeleteEventArgs args) {
            Console.WriteLine("Finishing app...");
            Application.Quit ();
        }
        static void hello (object obj, EventArgs args) {
            Console.WriteLine("Hello World!");
        }
}
```

Para compilar se utiliza el compilador de mono:

```
$ mcs hello.cs -pkg:gtk-sharp
```

Luego se puede ejecutar la pequeña aplicación:

```
$ mono hello.exe
```

Monodevelop: entorno de desarrollo para Mono y GNOME:

Monodevelop [13] es actualmente el entorno de desarrollo del proyecto Mono entre sus principales características se encuentra:

- Visor de Clases.
- Ayuda integrada al entorno.
- Completado de código.
- Soporte a proyectos.
- Depurador Integrado.

Una vista general de Monodevelop se puede observar en la Figura 3

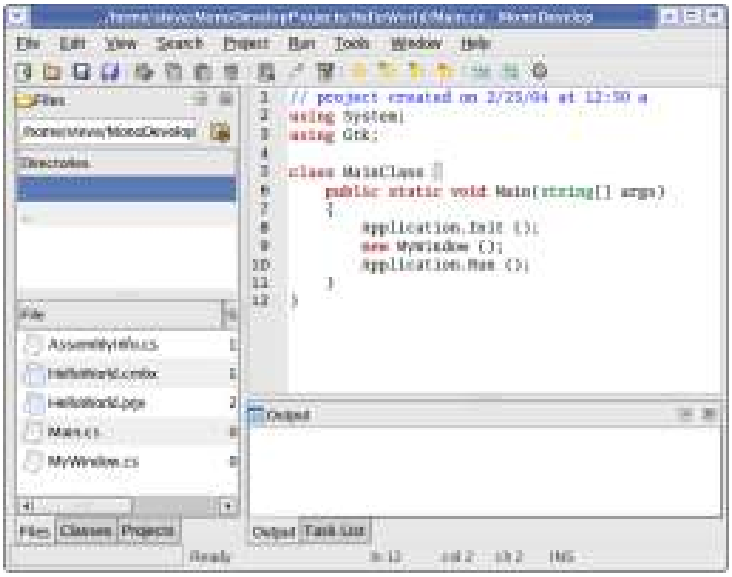

Figura 3 - Monodevelop

Monodevelop se encuentra aún en las faces iniciales de desarrollo, sin embargo ya es una herramienta potente que permite a desarrolladores con conocimientos en .NET poder migrar sus aplicaciones a otras plataformas como Linux, a su vez el desarrollo de aplicaciones de escritorio en GNOME pueden ser portadas a otros sistemas operativos. El sitio oficial de Monodevelop contiene gran cantidad de

tutoriales e información relacionada, además el entorno cuenta con ayuda integrada la cual muestra como comenzar su utilización desde los primeros pasos.

Python para GNOME

PyGTK [14] es un conjunto de módulos que componen una interfaz Python para GTK+, se encuentra soportado y distribuido por la mayoría de las distribuciones que incluyen GNOME. La simpleza del código escrito en Python hace de este proyecto una gran herramienta para el desarrollo de aplicaciones para el escritorio, además las características de multiplataforma de Python dan al desarrollador grandes ventajas al usar entornos con diversos sistemas operativos. Python es un lenguaje de programación interpretado, ampliable y orientado a objetos que se distribuye con un amplio conjunto de módulos que permiten el acceso a un gran número de servicios del sistema operativo, servicios de Internet (como HTML, XML, FTP, etc.), gráficos (incluidos OpenGL, TK, etc.), funciones de manejo de cadenas, servicios de correo (IMAP, SMTP, POP3, etc.), multimedia (audio, JPEG) y servicios de criptografía. Existen además multitud de módulos proporcionados por terceros que añaden otros servicios. Python se distribuye bajo términos similares a los de la licencia GPL y está disponible para los sistemas operativos Linux, Unix, Win32 y MacOS X.

Un ejemplo en Python usando PyGTK:

```
#!/usr/bin/env python
import gtk

class HelloWorld(gtk.Window):

    def hello(self, widget, data=None):
        print "Hello World!"

    def delete_event(self, widget, event, data=None):
        print "Finishing app..."
        gtk.main_quit()

    def __init__(self):
        gtk.Window.__init__(self, gtk.WINDOW_TOPLEVEL)
        self.set_default_size(150, 50)
        self.set_title("PyGTK DEMO")
        self.connect("delete_event", self.delete_event)
        self.button = gtk.Button("Hello!")
        self.button.connect("clicked", self.hello, None)
        self.add(self.button)
        self.show_all()

if __name__ == "__main__":
    hello = HelloWorld()
    gtk.main()
```

PIDA: entorno de desarrollo para Python y GNOME:

PIDA [15] (Python Integrated Development Application), es un IDE escrito en Python y PyGTK, su diferencia con otros proyectos del mismo tipo radica en que utiliza las herramientas que el desarrollador tiene disponibles, por lo tanto puede ser usado como un integrador de herramientas y utilidades. Aunque el proyecto es bastante joven en la actualidad muestra un interesante conjunto de características tales como:

- Autocompletado de código.
- Resaltado de Sintaxis.
- Administración de proyectos.
- Control de versiones integrado con CVS y Subversión.
- Depurador.
- Diseñador de interfaz integrado.
- Integración con Pastebin.

La principal característica que hace atractivo a PIDA es su sistema de complementos (plugins), el cual permite integrar herramientas de desarrollo dentro del entorno de trabajo, un par de ejemplos de esto es el uso del conocido editor de texto para UNIX vim y del diseñador de interfaz Gazpacho, los cuales cuentan con sus respectivos complementos. De esta manera PIDA puede ser extendido y configurado según las necesidades de cada proyecto. Una vista general se puede observar en la Figura 4.

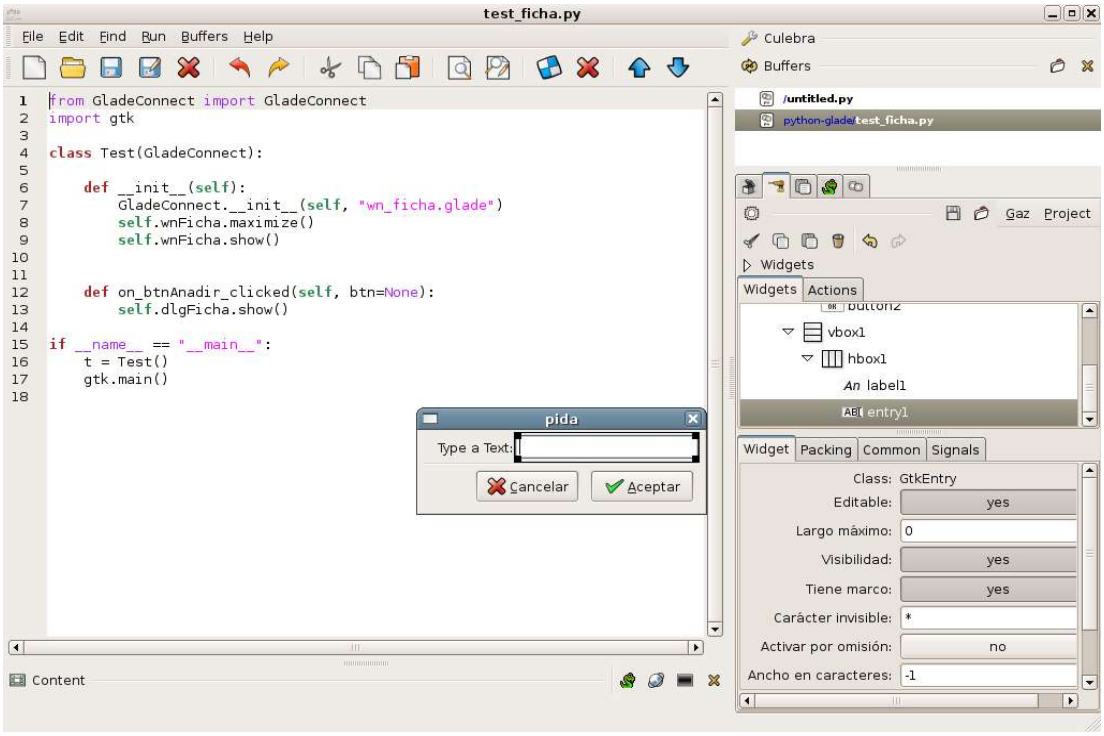

Figura 4 - PIDA

Conclusiones

En el presente artículo se exploran tres alternativas posibles para el uso de GNOME como plataforma de desarrollo, además se exponen las características multiplataforma de estas tecnologías. El uso de este tipo de tecnologías para el desarrollo rápido de aplicaciones al interior de empresas e instituciones así como dentro de la comunidad general es una posibilidad cierta más aún considerando que muchas de estas tecnologías son conocidas por desarrolladores de otras plataformas por lo que la curva de aprendizaje es mucho más baja, al eliminar algunas de las barreras de entrada.

Los proyectos presentados cuentan con un alto grado de madurez y el respaldo de importantes empresas e instituciones internacionales por lo que pueden ser considerados como alternativas válidas de desarrollo para proyectos al interior de las empresas.

Referencias

[1] GNOME, http://www.gnome.org
[2] GTK+, http://www.gtk.org
[3] Java, http://java.sun.com
[4] C#, http://www.ecma-international.org/publications/standards/Ecma-334.htm
[5] Python, http://www.python.org
[6] Java-GNOME, http://java-gnome.sourceforge.net/cgi-bin/bin/view
[7] Gnome Bindings Release, http://www.gnome.org/start/2.9/bindings
[8] Ubuntu Linux, http://www.ubuntulinux.org
[9] Eclipse, http://www.eclipse.org
[10] Fedora, http://fedora.redhat.com
[11] Mono, http://www.mono-project.com
[12] Gtk#, http://gtk-sharp.sourceforge.net/
[13] Monodevelop, http://www.monodevelop.com
[14] PyGTK, http://www.pygtk.org
[15] PIDA, http://pida.berlios.de/index.php/Main_Page