Desarrollo de discos virtuales usando el lenguaje Perl

Creation of virtual drives by means of Perl language

Manuel Dávila Sguerra

Ingeniero de Sistemas de la Universidad de los Andes. Director Depto de Informática Redes y Electrónica – Uniminuto, miembro de la red de Decanos y Directores de Ingeniería de Sistemas y Afines – REDIS; Miembro Fundador Indusoft y Acis, gerente de desarrollo Grupo Linux S.A.
mdavila@uniminuto.edu

Resumen

La necesidad de manejar repositorios de datos disponibles a cualquier hora y desde cualquier lugar hace que cada vez se necesiten más desarrollos de aplicaciones centradas en la web. Los discos duros de los pc's pierden sentido si los usuarios no los pueden acceder remotamente, funcionalidad que si da Internet. Este artículo está orientado a compartir la experiencia del desarrollo de un disco duro virtual dentro del proyecto e-Genesis – El generador de sistemas utilizando software libre y el lenguaje de programación Perl, entrando en algún detalle en el planteamiento del desarrollo, las tecnologías involucradas, la programación de prototipos, la implementación de facilidades de uso, el diseño de sistemas amigables, y la implementación de las funcionalidades.

Palabras clave. Discos duros virtuales, programación web, lenguaje Perl

Abstract

Managing data repositories available at any time and any place is a requirement that urges the development of web applications. PC hard drives become useless if users do not have remote access to them. This functionality is provided by the Internet. This paper is aimed to share the development experience of a virtual hard drive within the e-Genesis project. This is a software system generator that uses freeware and the Perl Programming Language. It will describe in detail the development approach, the integrated technologies, the programming of prototypes, the implementation to facilitate ease of use, the design of friendly systems, and the implementation of functionalities.

Keywords. Virtual hard drives, Web programming, Perl Language.

Introducción

El tema del presente artículo hace pensar en la palabra “ubiquidad” que según la real academia de la lengua se refiere a algo “que está presente a un mismo tiempo en todas partes” .

Las personas que usan computadores manejan muchos documentos con información de tipo documental, gráfico, audio visual y de muchas otras características que se almacenan en los discos duros de los computadores para su posterior utilización. Sin embargo esos documentos no le son útiles al usuario cuando se encuentran en sitios remotos sin acceso a sus pc’s lo cual ha hecho que los nuevos sistemas se orienten hacia el uso de una red, sin límites de hora ni de lugar, como lo es Internet.

El trabajo consistió en desarrollar una serie de programas que permitieran acceder a un servidor remoto con Linux para mantener de manera organizada los directorios en donde un usuario depositaría sus documentos con funcionalidades para su manejo y facilidades para su uso. En este documento se presentarán los prototipos del desarrollo que se diseñaron y se programaron como de base para la construcción de un sistema profesional. El objetivo buscado con este artículo es explicar la tecnología utilizada y el procedimiento para el desarrollo de este tipo de soluciones, no orientada al usuario final sino a los posibles desarrolladores de software e investigadores interesados en conocer la metodología utilizada en este caso.

Ambiente de trabajo

Siendo una aplicación para uso remoto se asume una conexión de Internet, como se puede observar en el Diagrama 1, en la cual se tienen varias tecnologías necesarias para cumplir el objetivo. Estas son: Sistema Operacional Linux. Servidor de web Apache, Interpretador de lenguaje Perl, protocolo de comunicaciones tcp/ip y conexión a Internet.

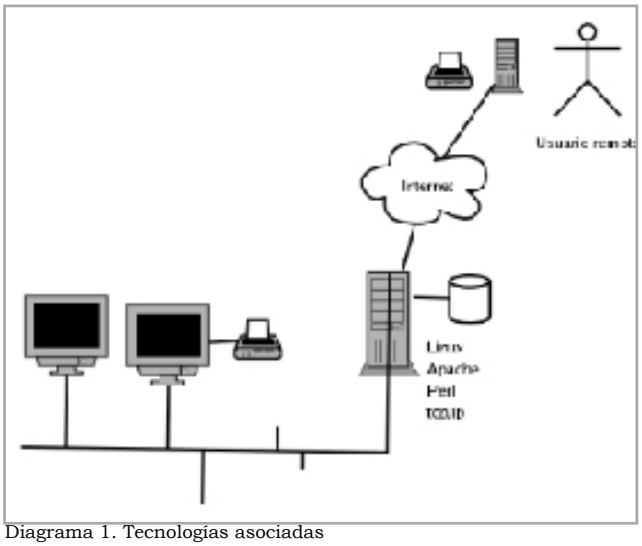

Diagrama 1. Tecnologías asociadas

La misma definición del trabajo hace entender que será necesario navegar dentro del disco duro del servidor para presentar los resultados de manera gráfica y textual dentro del navegador de un usuario remoto. El origen de dicha navegación son comandos que se ejecutan en el sistema operacional cuyos resultados deben ser capturados para crear la metáfora gráfica que de manera amigable los muestre presentando botones para activar las funcionalidades. El servidor debe ser accedido de manera remota utilizando un lenguaje, en este caso Perl basado en la ejecución de los comandos escogidos de Linux[1] como los que se presentan en la tabla 1.

Los CGI’s

El HTML o Hypertext Markup Language[2] es el lenguaje de marcas escogido para interactuar desde el navegador, presentar los resultados entregados por las aplicaciones. Este lenguaje de marcas tiene una estructura como se muestra en la tabla 2.

Ese texto almacenado utilizando un editor con la extensión .html producirá como resultado el mensaje “Hola mundo” en la pantalla del navegador al ser este orientado a http://linux.domain.com.co/pruebas/programa001.html

[1]Jack Tackett, Jr, David Gunter, Lance Brown. 1996. Linux Edición especial,
[2]Thomas A. Powell. 1998. Manual de Referencia HTML, McGrawHill,

Comando de Linux	Sintaxis	Efecto
archivo1 archivo2	Copiar archivos	Copia el archivo1 sobre el archivo2 dejando intacto el archivo1
d directorio	Cambio de directorio	Pasa al directorio especificado
ls -l	Lista el contenido de un directorio	Muestra los archivos y directorios que contiene incluyendo las propiedades como permisos y tamaños
pwd	Muestra el directorio actual en donde está posicionado el usuario o el programa	Útil para asegurar que se está en el sitio adecuado dentro del servidor
ls -F	Muestra los directorios que hay en la dirección en donde está posicionado el servidor	Facilita la identificación de directorios y archivos
mkdir	Crear un directorio	
mv archivo1 archivo2	Mueve el archivo1 a archivo2	Desaparece archivo1
rm archivo	Borra archivo	
rmdir	Borra un directorio	

Tabla 1. Ejemplos de comandos de Linux

En una aplicación como la mencionada es necesario generar páginas html de manera dinámica dependiendo de las acciones para las que se hayan diseñado los programas lo cual se hace usando CGI o Common Gateway Interface, mecanismo de software que permite que un programa se ejecuta en el servidor y genere hojas de html dinámicamente como se muestra en la tabla 3 para producir el mensaje "Hola mundo" en el navegador del usuario.

programa001.html: hoja hecha en html para escribir "Hola mundo" en el navegador

```
<html>
   <head>
      <title>Este es el título de la página</title>
   </head>
   <body>
      Hola mundo
   </body>
</html>
```

Tabla 2. Estructura del lenguaje de marcas html

programa002.pl: cgi hecho en perl que envía al navegador el mensaje Hola Mundo!!!

```
#!/usr/bin/perl
# Tipo MIME es requerido
print "Content-type: text/html\n\n";
print "<HTML>
        <HEAD>
          <TITLE>Titulo</TITLE>
        </HEAD>";
print "<BODY>
        Hola Mundo!!!
      </BODY>
</HTML>";
```

Tabla 3. Cgi para escribir "Hola mundo" usando Perl

El efecto de este programa es el mismo que se logra haciendo una página html, sólo que la de este ejemplo es generada de manera dinámica usando el lenguaje Perl. La instrucción print "Content-type: text/html\n\n"; es la que inicia la unidad estándar de salida en el navegador.

La forma de ejecutar un cgi se hace direccionando el navegador a una zona que maneja el servidor web en donde residen los programas. En el caso de Linux y Apache ese directorio se encuentra ubicado en /cgi-bin/pruebas/ siendo /cg-bin el directorio raíz para los cgi del servidor de web Apache[3]: http://linux.domain.com.co/cgi-bin/pruebas/programa002.pl

[3]Ben Laurie & Peter Laurie. 1999. Apache: The Definitive Guide, Second Edition, O'Reilly,

Desde los programas escritos en lenguaje Perl es posible controlar el comportamiento del servidor con la misma potencia de cuando eso se hace manualmente por parte de un administrador del sistema que se encuentre "sentado" frente a la consola. Inclusive acciones como hacer que se conozcan las variables de ambiente del sistema operacional es posible hacerlo como lo muestra el programa de la tabla 4.

programa003.pl: CGI para mostrar las variables de ambiente

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html>
        <body>
          <h1>Lista de variables de
ambiente</h1>";
foreach $variable (keys %ENV) {
  print "$variable $ENV{$variable}<br>"
}
print "</body></html>";
```

Tabla 4. Cgi para listar las variables de ambiente usando Perl

El resultado de direccionar el navegador a http://linux.domain.com.co/cgi-bin/pruebas/programa003.pl se ve en el Diagrama 2:

Diagrama 2. Resultado al listar las variables de ambiente

El embellecimiento de los resultados que se ven el navegador se logra con un adecuado uso del html y es una parte importante de resaltar en esta investigación porque de eso depende el diseño de sistemas amigables para beneficio del usuario final. Son muchos los elementos que se deben tener en cuenta como las fuentes de los textos, los fondos de las hojas proyectadas en el navegador, los tamaños y colores de los caracteres y muchos más aspectos que corresponden a la tarea de un diseñador gráfico o del mismo programador si tiene conocimiento del lenguaje de marcas y una capacidad de diseño y creatividad. El mismo programa anterior pero formateando los resultados dentro de una tabla del navegador se obtiene con un código como el que se muestra en la tabla 5.

programa004.pl: CGI para mostrar las variables de ambiente dentro de una tabla

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>
      <h1>Lista de variables de ambiente
formateada en una tabla</h1>";
print "<table border=1>";
foreach $variable (keys %ENV) {
  print "<td><h4>$variable</h4><td>
$ENV{$variable}<tr>"
}
print "</table></body></html>";
```

Tabla 5. Cgi que formatea los resultados en una tabla usando Perl

La instrucción print *"<table border=1>"; define una tabla con borde tamaño 1 y: print "<td><h4>$variable</h4><td> $ENV{$variable}<tr>"* lista la variable dentro de la fila correspondiente. Este cgi al ser ejecutado direccionando el navegador a http://linux.domain.com.co/cgi-bin/pruebas/programa004.pl muestra los resultados que se ven en el Diagrama 3:

Diagrama 3. Lista de las variables de ambiente formateada en una tabla

Estos prototipos fueron sirviendo para descubrir, paso a paso, la forma de lograr los resultados que fortalecieron la capacidad de programación y que, como es natural en la medida en que se avanza en un desarrollo de software, tomaría visos más complejos. La necesidad de llevar estadísticas de navegación en un sitio web es una aplicación típica en este tipo de desarrollos lo cual se hace implementando un contador de visitas escribiendo un programa (cgi) que defina un archivo residente en el servidor con un dato almacenado con el número de visitas al que cada vez que alguien entre a la página se le sume un uno y grabe el nuevo resultado en dicho archivo. El código fuente de este prototipo para el manejo de un "contador" se muestra en la tabla 6.

Programa005.pl: CGI para llevar un contador de visitas dentro de una página web

```
#!/usr/bin/perl
# Abrir el archivo que contiene el contador
open(CONT,"/var/www/cgi-bin/pruebas/conteo.txt");
# Leerlo
$cuenta = <CONT>;
# Aumentar el contador en uno
$cuenta++;
# Mostrarlo en la página web
print "Content-type: text/html\n\n";
print "<html><body>
    <h1>Esta pagina ha sido visitada $cuenta veces !!!</h1>";
print "</body></html>";
# Abrir el archivo de salida
open(CONT,">/var/www/cgi-bin/pruebas/conteo.txt");
# y grabarle el nuevo contador
print CONT $cuenta;
```

Tabla 6. Cgi para manejar un contador de visitas usando Perl

El resultado de su ejecución direccionado el servidor a http://linux.domain.com.co/cgi-bin/pruebas/programa005.pl se ve así en el Diagrama 4:

Diagrama 4. Salida de un programa contador de visitas

La programación de estos prototipos son parte de un modelo de desarrollo de software que permite ir avanzado de manera permanente mientras se asimila la tecnología y permite prepararse para resolver problemas más y más complejos que correspondan al diseño del sistema. En el caso del disco virtual este se basa en un programa que ejecute comandos del sistema operacional Linux, como ya se había mencionado, desde un programa usando Perl, es decir un cgi, de tal manera que una vez obtenidos los resultados se "atrapen" en variables del programa para poderlos mostrar en el navegador, con las funcionalidades diseñadas para su manejo. Los comandos que se escojan para ser ejecutados dependen del diseño del sistema y del nivel de funcionalidades que se vayan a implementar de tal manera que continuando con la programación de prototipos en la tabla 7 se muestra el código fuente que hace lo pertinente ante la ejecución de un comando que lista el contenido de un directorio.

programa006.pl: lista simple de los archivos del sistema

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>
    <h1>Lista simple de Archivos</h1>";
# Define la tabla
print "<table border=1>";
$directorioalistar = ".";
# Ejecuta el comando en el directorio .
open (DIR, "ls -l $directorioalistar|");
#  se el resultado sobre esta variable
@directorioestado = <DIR>;
close (DIR);
# Recorre el listado y lo muestra
foreach $d (@directorioestado) {
    print "<tr><td>$d</td></tr>"
}
print "</table></body></html>"
```

Tabla 7. cgi para listar el contenido de un directorio usando Perl

La ejecución de http://linux.domain.com.co/cgi-bin/pruebas/programa006.pl desde el navegador da como resultado lo que muestra el diagrama 5:

Diagrama 5. Lista simple de un directorio de un disco duro del servidor

Para lograr sistema amigables se usan iconos que faciliten la visualización de la información por lo que se hace conveniente en el prototipo diseñar galerías gráfica que crecerán en la medida en que se implementen los programas definitivos y que dependen del diseño gráfico del sistema. Que sirva de ejemplo para este caso el diseño de un icono llamado folder.gif representado por la imagen que se ve en el Diagrama 6.

Con el uso de este icono el programa prototipo para listar los archivos mejora la presentación visual como se ve en el código mostrado en la tabla 8.

Diagrama 6. Icono representativo de la existencia de un archivo

Al ejecutar el programa direccionando el navegador a http://linux.domain.com.co/cgi-bin/pruebas/programa007.pl da como resultado lo que se muestra en el Diagrama 6.

Diagrama 6 - diagrama 5. Lista amigable de un directorio de un disco duro del servidor

programa007.pl: lista los archivos del sistema con icono

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>
   <h1>Lista de Archivos con Iconos</h1>";
# Define una tabla
print "<table border=1>";
$directoriolistar = ".";
# Ejecuta el comando en el directorio .
open (DIR, "ls -l $directoriolistar|");
# Lee el resultado sobre esta variable
@directoriolistado = <DIR>;
close (DIR);
# Recorre el listado y lo muestra
foreach $d (@directoriolistado) {
   print "<tr><td><h5><img SRC=../../pruebas/folder.gif BORDER=0 height=16 width=16></td><td>$d</td></h5></tr>";
}
print "</table></body></html>";
```

Tabla 8. Cgi que utiliza iconos para mejorar la visualización

```
programa000.pl: lista los archivos del sistema con icono y un enlace web

#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>
   <h1>Lista de Archivos con enlace web</h1>";
# Define una tabla
print "<table border=1>";
$directorio = ".";
# Ejecuta el comando en el directorio
open (DIR, "ls -l $directorio|");
# Lee el resultado sobre esta variable
@directoriolistado = <DIR>;
close (DIR);
# Recorre el listado y lo muestra
foreach $d (@directoriolistado) {
   print "<tr><td><h5><a href=\"http://linux.domain.com.co/cgi-bin/pruebas/programa009.pl\"><img SRC=../../pruebas/folder.gif
title=\"Enlace hacia una funcionalidad\" ALT=\"Enlace hacia una funcionalidad\" BORDER=0 height=16
width=16></td><td>$d</td></h5></tr>";
}
print "</table></body></html>";
```

Tabla 9. Un enlace web

Funcionalidades

Las funcionalidades del disco duro virtual, es decir las acciones que se programan para usarlo eficientemente, se fueron desarrollando a través de cgi's que se enlazan desde un interfaz cuyo diseño es parte del trabajo, haciendo uso de enlaces html que activen programas en el servidor. Se denomina un enlace el uso de un hipertexto que permite encadenar una hoja html o un cgi desde una página web dinámica o estática para definir el rumbo de un programa. La manera de enlazar bien sea un texto o un icono hacia una funcionalidad se muestra en el programa que se muestra en la tabla 9 el cual crea un enlace web desde el icono hacia un programa que produce un resultado.

En el programa se enlaza el icono, como se puede observar en el listado, introduciendo la siguiente marca de html:

```
print "<tr><td><h5><a
href=\"http://linux.domain.com.co/cgi-
bin/pruebas/programa008.pl\"><img
SRC=../../pruebas/folder.gif
title=\"Enlace hacia una funcionalidad\"
ALT=\"Enlace hacia una
funcionalidad\" BORDER=0 height=16
width=16></td><td>$_</td></h5></tr>"
;
```

En esta instrucción se puede observar que "a href" encadena hacia el programa009.pl que es el que se ejecutará si se direcciona el navegador hacia: http://linux.domain.com.co/cgi-bin/pruebas/programa008.pl que al ejecutarlo se ve el resultado como lo muestra el Diagrama 7:

Diagrama 7 Lista amigable de un directorio de un disco duro del servidor con un enlace web

La diferencia de este programa con el anterior radica en que el icono enlaza hacia un cgi que implementa la funcionalidad que se activa al darle click al icono enlazado, cgi que produce en este caso un mensaje elemental como se observa en el, Diagrama 8:

Diagrama 8. Resultado de una funcionalidad accedida desde un icono con enlace web

El caso anterior es elemental ya que llama un programa que sólo produce un mensaje, pero para el desarrollo de funcionalidades avanzadas se necesita enviar al programa las variables que determinarán la funcionalidad que gobernarán el comportamiento del programa llamado.

El prototipo siguiente mostrado en la tabla 10 hace que el programa010.pl llame al programa

La instrucción

print "<tr><td><h5><a href=\"http://linux.domain.com.co/cgi-bin/pruebas/programa009.pl\"><img SRC=../../pruebas/folder.gif title=\"Enlace hacia una funcionalidad\" ALT=\"Enlace hacia una funcionalidad\" BORDER=0 height=16 width=16></td><td>$_</td></h5></tr>";

determina el llamado al programa que ejecutará la funcionalidad. El nombre del directorio que deberá listar el programa llamado, se define en ?fdirectorio=directorio como se observa en la instrucción y que aparece después de especificar la llamada al programa011.pl

En otras palabras el programa011.pl sabrá que debe listar los archivos del directorio mencionado.

programa010.pl: lista los archivos con un enlace cgi es decir con un programa que tiene una funcionalidad, en este caso llama al programa011.pl el cual lista el directorio especificado por este programa. El nombre del directorio es "directorio"

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><body>
   <h1>Lista de Archivos con enlace web más elaborado</h1>";
# Define una tabla
print "<table border=1>";
$directorio=listar = ".";
# Ejecute el comando en el directorio .
open (DIR, "ls -l $directorio=lister|");
# lea el resultado sobre esta variable
@directorio=listado = <DIR>;
close (DIR);
# Recorre el listado y lo muestra
foreach $d (@directorio=listado) {
   print "<tr><td><h5><a href=\"http://linux.domain.com.co/cgi-bin/pruebas/programa011.pl?fdirectorio=directorio\"><img
SRC=../../pruebas/folder.gif title=\"Enlace hacia una funcionalidad\" ALT=\"Enlace hacia una funcionalidad\" BORDER=0 height=16
width=16></td><td>$d</td></h5></tr>";
}
print "</table></body></html>";
```

Tabla 10. Enlaces web pasando datos al programa llamado

011.pl haciendo que éste liste los archivos de un directorio llamado "directorio" cuyo nombre es pasado por el cgi al navegador[4] .

Para ejecutar este programa se debe direccionar el navegador a http://linux.domain.com.co/cgi-bin/pruebas/programa010.pl y el efecto es el que se muestra en el diagrama 9.

Como los iconos que determinan la presencia de un archivo están enlazados, al poner sobre ellos el ratón del computador aparecerá la mano que indica la presencia de un enlace que al seleccio-

[4]Clinton Wong. 1997. Client Programming with Perl Automating Tasks on the Web 1st Edition, O'Reilly,

programa011.pl: lista los archivos del directorio especificado en el programa que lo llama es decir programa010.pl

```
#!/usr/bin/perl
$| = 1;
# Determina la unidad de salida
print "Content-type: text/html\n\n";
# Blanquea el arreglo asociativo
# que contendrá las variables pasadas
# por el programa que llama
%campos = ();
# Llama la función que lee las variables
ObtenerDatosForma(*%campos);
print "<html><body>
   <h1>Lista de Archivos producto de una <br>funcionalidad para listar un directorio</h1>";
# Define una tabla
print "<table border=1>";
# Ejecuta el comando ls -l en el directorio
open (DIR, "ls -l $campos{'directorio'}|");
@directoriolistado = <DIR>;
close (DIR);
# Recorre los datos leídos
foreach $d (@directoriolistado) {
   print "<tr><td><h5><a href=\"http://linux.domain.com.co/cgi-bin/pruebas/programa009.pl?><img SRC= ../../pruebas/folder.gif
title=\"Enlace hacia una funcionalidad\" ALT=\"Enlace hacia una funcionalidad\" BORDER=0 height=18
width=18></a></td><td>$d</td></h5></tr>";
}
print "</table></body></html>";
#--------------------------------
#            ObtenerDatosForma
#       Rutina para pasar los datos
#       de la forma a memoria
#--------------------------------
sub ObtenerDatosForma
{
   my($hashRef) = shift;
   my($buffer) = "";

   if ($ENV{'REQUEST_METHOD'} eq "GET")
      $buffer = $ENV{'QUERY_STRING'};
   }
   else {
      read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
   }
   foreach (split(/&/, $buffer)) {
      my($llave, $valor) = split(/=/, $_);
      $llave = decodificarURL ($llave);
      $valor = decodificarURL ($valor)
#     convierte multiples tags <P>
      $valor =~ s/(<P>\s*)+/<P>/ig;
#     escape tags HTML
      $valor =~ s/</&lt;/g;
      $valor =~ s/>/&gt;/g;
#     escape tag bold
      $valor =~ s/&lt;b&gt;/<b>/ig;
      $valor =~ s/&lt;\/b&gt;/<\/b>/ig;
#     preserve tag caliente
      $valor =~ s/&lt;b&gt;/<b>/ig;
      $valor =~ s/&lt;\/b&gt;/<\/b>/ig;
#     remueve carriage returns innecesarios
      $valor =~ s/\cM//g;
#     convierte 2 newlines en un parrafo
      $valor =~ s/\n\n/<P>/g;
#     convierte newline en espacio
      $valor =~ s/\n/ /g;
      $$hashRef{$llave} = $valor;
   }
}
sub decodificarURL {
   $_ = shift;
   s/\+/ /;
   s/%(..)/pack('c', hex($1))/eg;
   return($_);
}
```

Tabla 11. Mecanismo para que el programa llamado "lea" las variable que pasa el programa de origen.

Diagrama 9. Resultado de llamar un programa que enlazará una funcionalidad más elaborada

narlo con el ratón, activará al programa11.pl que será el encargado de ejecutar la orden, que en este caso es la de listar los archivos del directorio. El código fuente se ve en la tabla 11 (página anterior).

En este prototipo se ha implementado una función llamada ObtenerDatosForma que parece un poco compleja de entender pero que es un estándar para leer las variables que el programa de origen pasa a través del cgi.

El resultado al seleccionar el enlace sobre el icono se observa en el Diagrama 10.

Diagrama 10. Resultado de ejecutar el programa que lleva a cabo una funcionalidad.

Facilidad de uso

Estos prototipos y otros más fueron la base para la construcción completa de un disco duro virtual utilizando los mecanismos de programación explicados y combinándolos con un buen uso del lenguaje de marcas html y Perl para lograr un sistema completo.

La combinación entre la facilidad de uso y la funcionalidad determinan la capacidad del sistema y son muchas las variaciones que se pueden presentar. No hay aun un estándar definido y está en la creatividad y capacidad del programador la implementación de las mismas.

Se muestra a continuación un ejemplo de un disco virtual implementado en el sistema e-genesis-El generador de sistemas.

En este caso el icono se usó para la presencia de directorios y las funcionalidades implementadas son:

- **Nuevo.** Para crear un nuevo directorio. Al seleccionarla se abre una ventana que pide el nombre de la nueva carpeta o directorio.
- **MiRed Externa.** Es una funcionalidad que permite navegar entro de un pc normal de la red local desde Internet.
- **Seleccionar.** Esta columna permite selec-

cionar el directorio que se quiere abrir para ver sus archivos y sub carpetas, sobre las cuales se pueden bajar o subir archivos como se desee.

Si se abren los directorios se pueden ver de esta forma:

Las funcionalidades que da esta ventana son:

- Subir. Para subir un archivo desde el Pc hasta el servidor del disco virtual al directorio que está abierto en ese momento. El sistema le mostrará un botón para navegar sobre el pc y buscar el archivo que se desea subir.
- Nuevo. Para abrir un nuevo directorio. El sistema le pedirá el nombre del directorio.
- Bajar. Para bajar archivos desde el disco duro virtual al pc.
- Borrar. Para borrar archivos seleccionados.
- Mover. Para mover archivos a otros directorios.
- Copiar. Para copiar archivos a otros directorios.

En la parte de abajo de la ventana se puede solicitar unos servicios como comprimir directorios y pedir que se listen sus propiedades

Conclusiones

El uso de prototipos en el proceso de la programación de sistemas constituye un cambio con respeto de la vieja tradición de hacer primero diseños muy sofisticados y luego entrar a la etapa de la programación. Hoy en día los sistemas deben interactuar con las plataformas tecnológicas para crear funcionalidades que le sirvan a los usuarios para manejar las opciones con más facilidad..

Se ha propuesto explicar el método que se siguió para el diseño y programación de un disco virtual orientado a la web utilizando software libre y el lenguaje de programación Perl a través de una serie de prototipos que permitieron aprender e implementar las bases de la programación usando cgi's que conectan programas en la web y comprobar el funcionamiento de opciones que más adelante se fueron afinando en la medida en que se implementó el sistema final.

Para iniciar el proyecto fue necesario conocer los fundamentos del lenguaje de marcas html y la manera de crear páginas web de manera dinámica usando el lenguaje Perl para la generación dinámica de páginas que permitieron implementar las funcionalidades del disco duro virtual.

Paso a paso, se fue aprendiendo a utilizar el lenguaje y manejar desde los programas la tecnología de base, en este caso el sistema operacional Linux, construyendo programas prototipo cuya complejidad final se logró en un proceso continuado de pruebas y afinamientos.

La implementación final fue entonces el resultado de un proceso de investigación aplicada con el lenguaje, el acceso a la plataforma de base, la implementación de las funcionalidades diseñadas de antemano para lograr los resultados que se pretendieron en la formulación del problema.

Los discos duros virtuales hoy en día aparecen en todos los sistemas orientados al trabajo en grupo, se complementan con opciones de seguridad para que los usuarios puedan definir los permisos que tendrán los participantes del grupo

para crear, modificar, leer, borrar, copiar y mover archivos dentro del servidor y de manera remota.

Finalmente, se demostró que este tipo de aplicaciones son factibles de ser construidas localmente invitando a continuar con un proceso que nos convierta en fabricantes de software y no sólo en simples consumidores de tecnologías.

Referencias bibliográficas

Laurie, Ben & Peter Laurie. 1999. Apache: The Definitive Guide, Second Edition, O'Reilly.

McDaniel, Robert 1996. CGI Manual of Style, Macmillan Computer Publishing USA

Powell, Thomas A.. 1998. Manual de Referencia HTML, McGrawHill,

Tackett, Jack Jr. David Gunter, Lance Brown. 1996. Linux Edición especial,

Till, David. 1996. Teach yourself Perl 5, Sams publishing.

Wong, Clinton. 1997. Client Programming with Perl Automating Tasks on the Web 1st Edition, O'Reilly.