AÑO ---------- 0
NÚMERO -------- 3
FECHA: 2013-01-28

#3

"ELVIS"

HD Hackers & DEVELOPERS

Magazine digital de distribución
mensual sobre Software Libre, Hacking y Programación
para profesionales del sector de Tecnologías de la Información

Staff

Eugenia Bahit — Arquitecta GLAP & Agile Coach
Indira Burga — Ingeniera de Sistemas
Laura Mora — Administradora de Redes y Sistemas
María José Montes Díaz — Técnica en Informática de Gestión
Milagros Infante Montero — Est. Ingeniería de Sistemas

HD Hackers & DEVELOPERS #3

Hackers & Developers Magazine, es una iniciativa sin fines de lucro destinada al fomento y difusión de las tecnologías libres presentes o futuras, bajo una clara óptica docente y altruista, que resulte de interés técnico y/o científico a profesionales del sector de Tecnologías de la Información. Hackers & Developers Magazine se sostiene económicamente con el apoyo de la comunidad, no recibiendo subvención alguna de ninguna empresa, organización u organismo de Gobierno. Necesitamos de tu apoyo para poder mantener este proyecto.

Ayúdanos a continuar con este proyecto

Puedes hacer un donativo ahora, de 10, 15, 25, 50, 100 o 150 USD para ayudar a que Hackers & Developers Magazine pueda seguir publicándose de forma gratuita, todos los meses. Puedes donar con PayPal o Tarjeta de Crédito a través del siguiente enlace:
www.hdmagazine.org/donar

CON TU DONACIÓN DE USD 150
RECIBES DE REGALO, UNA FUNDA DE NEOPRENE PARA TU ORDENADOR PORTÁTIL
VALUADA EN USD 25.-
(Origen: Estados Unidos)

"Hacker es alguien que disfruta jugando con la inteligencia"
Richard Stallman
Free Software, Free Society
(Pág. 97), GNU Press 2010-2012

En esta edición:

Y LAS SECCIONES DE SIEMPRE:

Créditos

Hackers & Developers Magazine es posible gracias al compromiso de:

Responsable de Proyecto
Eugenia Bahit

Responsables de Comunicación

Indira Burga (Atención al Lector) - Milagros Infante (Difusión)

Staff Permanente

Eugenia Bahit
Arquitecta GLAMP & Agile Coach
www.eugeniabahit.com

Indira Burga
Ingeniera de Sistemas
about.me/indirabm

Milagros Infante Montero
Estudiante de Ingeniería en Sistemas
www.milale.net

Laura Mora
Administradora de Redes y
Sistemas GNU/Linux
blackhold.nusepas.com

María José Montes Díaz
Técnica en Informática de Gestión
archninfa.blogspot.com.es

Colaboradores Estables

Elizabeth Ramírez
(Ingeniera Electrónica)

Sergio Infante Montero
(Ingeniero Informático)

Yecely Díaz
(Maestra en Inteligencia Artificial)

Redactores Voluntarios

Celia Cintas
Eliana Caraballo

Difusión

Hackers & Developers Magazine agradece a los portales que nos ayudan con la difusión del proyecto:

www.debianhackers.net

www.desarrolloweb.com

www.desdelinux.net

E-mail de Contacto:

contacto@hdmagazine.org

Twitter Bootstrap: un elegante, intuitivo y poderoso framework

¿Buscas un framework que te permita hacer magia en tus desarrollos? Conoce a Twitter Bootstrap y sorpréndete con la cantidad de cosas geniales que puedes hacer. Cómo el *Responsive Web Design* combinado con esta herramienta te da a entender gráficamente como puedes obtener un diseño para gobernar a todos.

Escrito por: **Milagros Alessandra Infante Montero** (Est. Ing. Informática)

Estudiante de Ingeniería Informática. Miembro de **APESOL** (Asociación Peruana de Software Libre) y de la comunidad de software libre **Lumenhack**. Miembro del equipo de traducción al español de **GNOME**. Apasionada por el desarrollo de software, tecnología y gadgets. Defensora de tecnologías basadas en software libre y de código abierto.

Webs:
Blog: www.milale.net

Redes sociales:
Twitter / Identi.ca: **@milale**

En los primeros días de Twitter, se usaban muchas librerías que fueran familiares con los requerimientos del *front-end* pero había inconsistencias que dificultaban el poder mantenerlos; y es cuando Bootstrap nació y creció significativamente.

¿Por qué Twitter Bootstrap?

Al inicio Bootstrap solo era CSS, pero a medida que pasó el tiempo ganó un rango de características como declaraciones identadas, variables, operaciones y funciones de

color. Las ventajas más resultantes que tiene es su fácil implementación con solo ponerlo en tu código y el poderoso CSS que contiene y cubre las necesidades del desarrollo de la web.

Al ver en profundidad a Bootstrap encontraremos 7 archivos diferentes:

1. **reset.less:** Un restablecimiento CSS creado por Eric Meyer y modificado para nuestro uso eliminando elementos innecesarios.
2. **preboot.less:** Variables de color y *mixins* para gradientes, transparencias y transiciones.
3. **scaffolding.less:** Estilos básicos y globales para generar un sistema de red, diseño estructural y plantillas de página.
4. **type.less:** Cabeceras, texto del cuerpo, listas, código y un sistema de tipografía versátil y durable.
5. **patterns.less:** Elementos de interfaz repetibles como navegación, modales e información sobre herramientas para llevarlo más allá de los estilos predeterminados.
6. **forms.less:** Estilos duraderos para diversos tipos de entrada, diseño de formularios y estados de control.
7. **tables.less:** Estilos para datos tabulares en pantallas variadas.

Bootstrap funciona dando una solución clara y uniforme a las tareas de de interfaz de cada día de los desarrolladores; se ha convertido en una de las muchas herramientas del *front-end* en nuevas aplicaciones y sitios Web. Se usa para arrojar prototipos rápidos y guía la ejecución de diseños mas sofisticados y esfuerzos de ingeniería más grandes. Es el camino simple para aplicaciones altamente *usables*, limpias y rápidas.

Responsive web design con Twitter Bootstrap

Responsive web design es un enfoque, el cual debe cambiar tu manera de pensar. La idea básica detrás es: un diseño para gobernarlos a todos (sí, como en el Señor de los anillos, un anillo para gobernar a todos).

No m.tu-dominio.com; no touch.tu-dominio.com; no 3 archivos de CSS separados; no 7 archivos gráficos para cada dispositivo o cada orientación; solo “tu-dominio.com” tal cual en plataformas de escritorio, tablets, Smart Phones, etc.

“One design to rule them all”

Twitter Bootstrap ofrece un montón de cosas increíbles:

- Estilos globales para el cuerpo para restablecer tipo y fondo, estilos de enlace,

sistema de red y dos diseños simples.

- Estilos para elementos HTML comunes como tipografía, código, tablas, formularios, botones, un conjunto de iconos, etc.
- Estilos básicos para componentes de interfaz comunes como pestañas, alertas, cabeceras de página y más.
- Plugins de JavaScript para cosas como información sobre herramientas, modales, etc.
- La más importante, es que es fácil de aprender y usar, ya que tiene muy buena documentación y todos los ejemplos con los que un desarrollador puede soñar.

Empezando con Twitter Bootstrap

Al entrar a la página oficial[1] descargamos la versión personalizada y luego de extraer el paquete obtenemos lo siguiente:

```
milagros@joe:~code/bootstrap$ tree
.
├── css
│   ├── bootstrap.css
│   ├── bootstrap.min.css
├── js
│   ├── bootstrap.js
│   ├── bootstrap.min.js
├── img
│   ├── glyphicons-halflings.png
│   └── glyphicons-halflings-white.png
└── prueba.html
```

Para descargar Twitter Bootstrap, ingresa en http://twitter.github.com/bootstrap/

Por ejemplo, si tenemos un archivo HTML en blanco:

```
<!DOCTYPE html>
<html>
    <head>
          <meta charset="utf-8">
          <title>Proyecto sin bootstrap</title>
    </head>
    <body>
          <p>Nuestra página aún no es responsive</p>
    </body>
</html>
```

Para tener ya Bootstrap en nuestro archivo debemos hacer la debida referencia al CSS y

1 http://twitter.github.com/bootstrap/

al JS de Bootstrap:

```
<!DOCTYPE html>
<html>
    <head>
          <meta charset="utf-8">
          <title>Proyecto con bootstrap</title>
          <link rel="stylesheet" href="css/bootstrap.min.css">
          <script src="js/bootstrap.min.js"></script>
    </head>
    <body>
          <p>Nuestra página aún no es responsive</p>
    </body>
</html>
```

Layouts en Twitter Bootstrap

Diseño Fijo

Debes elegir esta opción si estás creando un sitio web estándar y probablemente no necesites el 100% de tu pantalla. Se elige la opción 940px. Provee un diseño de anchura fija (y opcionalmente *Responsive*) solo requiriendo de `<div class="container">`.

```
<body>
    <div class="container">
    ...
    </div>
</body>
```

Diseño Fluido

Esta opción es por si necesitas que tu aplicación use todo el ancho (el 100%) de tu pantalla. Crea un diseño fluido: una página de dos columnas con `<div class="container-fluid">`, ideal para aplicaciones y documentos.

```
<div class="container-fluid">
    <div class="row-fluid">
        <div class="span2">
            <!--Sidebar content-->
        </div>
        <div class="span10">
            <!--Body content-->
        </div>
    </div>
</div
```

Veamos la magia de Bootstrap...

Por ejemplo, vamos a usar un layout fijo:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Empezando con Bootstrap en Hackers and Developers Magazine</title>
    <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css"">
</head>
<body>

<div class="container">
    <div class="hero-unit">
        <h1>Ejemplo de diseño responsive para Hackers and Developers
            Magazine</h1>
        <p>Hola lectores de Hackers and Developers Magazine, gracias por
            seguirnos, aquí pueden contactarse con nosotras.</p>
        <p><a class="btn btn-primary btn-large">Conócenos &raquo;</a></p>
    </div>

    <div class="row">
        <div class="span4">
            <h2>Redactores</h2>
            <p>Conoce aquí a las personas detrás de este proyecto: Quienes
                forman parte del staff, quienes nos brindan su apoyo como
                redactores voluntarios y más. </p>
            <p><a class="btn" href="#">Pulsa aquí &raquo;</a></p>
        </div>

        <div class="span4">
            <h2>Artículos</h2>
            <p>Aquí encontrarás la lista de temas ya tratados en H&D Magazine,
            pulsa para verlos y encontrar información muy útil. </p>
            <p><a class="btn" href="#">Pulsa aquí &raquo;</a></p>
        </div>

        <div class="span4">
            <h2>Lineamientos</h2>
            <p>Hackers & Developers Magazine, surge como una iniciativa sin
            fines de lucro destinada al fomento y difusión de las tecnologías
            libres presentes y futuras, bajo una clara óptica docente que
            resulte de interés técnico y/o científico. </p>
            <p><a class="btn" href="#">Pulsa aquí &raquo;</a></p>
        </div>
    </div>
</div>

</body>
</html>
```

Descomponiendo el código veremos:

- `<div class=”container”></div>` Esto sirve como contenedor para todo el contenido de tu página, en caso de que se quiera usar un *layout* fluido solo se añade como clase, `container-fluid` en vez de `container`.
- `<div class=”row”></div>` Esto sirve para crear filas, es un contenedor de columnas que cuida los márgenes, relleno, etc. En caso de un *layout* fluido solo se reemplaza por `row-fluid`.
- `<div class=”span4”></div>` Un *layer* con la clase *span* es una columna (el máximo es 12 columnas); en el ejemplo existen 3 cajas en una fila (span4, porque 12/3 es

4. Si se quisieran mostrar 6 columnas, 12 / 6 = span2).

- `<div class="hero-unit"></div>` Esto es otro componente de Bootstrap añadido para hacerlo más genial.

Pero este ejemplo no debe quedar ahí.

Al revisar la documentación te encontrarás con componentes de interfaz de usuario más avanzadas. También puedes añadir una serie de *plugins* de JavaScript estandarizados, como pestañas o controles deslizantes que van muy bien con el estilo de Bootstrap predeterminado.

Y nuestro resultado será:

Y lo genial de todo esto es cuando eligiendo el diseño fluido (container-fluid y row-fluid) *redimensionas* tu navegador y te das cuenta de que todo es *responsive* y así es como puedes empezar a hacer magia con el elegante, intuitivo y poderoso Twitter Bootstrap.

"La simplicidad es la máxima expresión de la sofisticación"

Web Scraping: excavando en la red

Utilizados en ciencia ficción por Trish Dunne en su araña de búsqueda para encontrar el SYMBOLON, la construcción de web scrapers y spiders se ha convertido en el nuevo FizzBuzz a la hora de presentar una prueba técnica en una entrevista de trabajo, por lo que no está de más conocer las técnicas básicas de su funcionamiento.

Escrito por: **Elizabeth Ramírez** (Ingeniera Electrónica)

Desarrolladora de Software en las Industrias Web, Telecomunicaciones y Entretenimiento. Especialista en Sistemas de Información, con énfasis en Real-Time Billing. Miembro de IEEE Signal Processing Society, New York Section. "Wantrepreneur".

Webs:
About.Me: http://about.me/elizabethr

Redes sociales:
Twitter: **@eramirem**

En algunas ocasiones, hemos necesitado procesar información de la web de una manera diferente a la convencional, es decir, usando un navegador. El navegador sólo descarga y *renderiza* datos de páginas web, pero si tenemos necesidades específicas de búsqueda de información, de ejecución de acciones y/o toma de decisiones usando la información encontrada, debemos hacerlo de manera manual.

Cuando descargamos el contenido de una página web usando un navegador, lo realizamos de manera manual y, a partir de allí debemos decidir qué información de la descargada es relevante para nuestra necesidad, también de manera manual. Los *web scrapers* y *spiders* permiten hacer esta búsqueda, descarga y procesamiento de información de manera programática y automática.

Los web scrapers ofrecen automatización y cierto nivel de inteligencia para la obtención y procesamiento de datos de internet. Podría decirse que los web scrapers nos sirven para descubrir el "internet oculto" en la capa de presentación de datos. Es básicamente adaptar la *World Wide Web* a necesidades personales y capitalizar información que está "escondida" en la web. Entre los diferentes usos para un web scraper se pueden encontrar:

- Agregación de noticias que solo muestre historias nuevas relevantes, ignore las historias redundantes y las que hayan sido leídas previamente.

- Detectar mercancías/servicios al más bajo precio, típico en los sitios que ofrecen reserva de hoteles y tarifas de avión.
- Optimización de páginas web, permitiendo analizar la estructura del HTML y el CSS, optimizándola para móvil.
- Cuando un sitio web despliega información acerca de un tópico de nuestro interés, pero no proporciona una API para que nuestros propios programas realicen consultas de dicha información, la solución es usar *screen scraping*. Por ejemplo: información de la bolsa de valores, tasas de cambio, eventos.

> *Sin demasiado optimismo, activó la araña, dando así inicio a una partida mundial de go fish. A velocidad cegadora, la araña se puso a comparar las frases con textos de todo el mundo…, en busca de un equivalente exacto. Dan Brown – El Símbolo Perdido.*

Normalmente, los conceptos de *crawler*, *scraper* y *spider*, son usados de manera indistinta. Aunque no existe una definición precisa para estos términos, hay algunas diferencias en su funcionamiento y los casos en que son usados.

Un *web crawler* recorre enlaces en la web usando un sitio de partida y permite crear copias del contenido de los sitios visitados, de manera similar a un motor de búsqueda. El proceso de *spidering* es el que permite iterar a través de los enlaces encontrados en el proceso de crawling hasta el nivel de profundidad indicado. Los enlaces son identificados mediante las etiquetas <a> de cada página, por lo que es requerido un análisis sintáctico (parsing[2]) del HTML. La combinación del proceso de crawling + spidering conforma el conocido *web robot*. El web robot debe regirse por lo especificado en el archivo robots.txt[3], evitando visitar los sitios que no están autorizados en dicho archivo.

Un web scraper en cambio, realiza la extracción de información de sitios específicos, buscando expresiones regulares, palabras clave, elementos, atributos, entre otros. Para llevar a cabo la automatización de la búsqueda y procesamiento de información en la web, los tres procesos trabajan en conjunto.

Anatomía

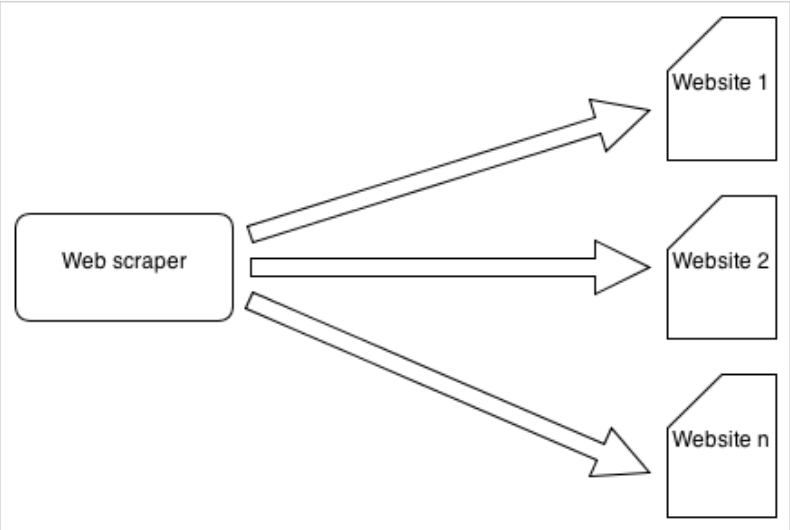
Existen varios esquemas de operación de un web scraper, entre los más básicos se encuentran los siguientes:

- **Uno-a-muchos**

  La estructura de uno a muchos es común de un scraper que busca por ejemplo,

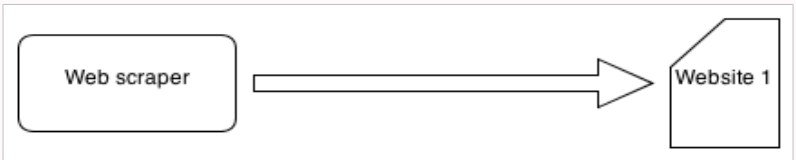
2 Separación de la información que necesitamos, la cual está contenida en otro conjunto de datos
3 *The Robots Exclusion Protocol*. Para más información, visitar http://www.robotstxt.org/robotstxt.html

información de precios de boletos de avión a través de diferentes sitios.



- **Uno-a-uno**

  Esta estructura es más típica de un *screen scraper*, en el cual se accede al contenido de un sitio web específico. Por ejemplo, buscar el "top 5" de historias con mayor calificación en un blog.



Cuando se escribe un web scraper, se debe tratar de imitar el comportamiento humano al usar un navegador, pero de manera programática. Introducir *delays* aleatorios -o al menos variables (no realizar el scraping exactamente cada 30 minutos, por ejemplo)- y no lanzar las tareas de scraping en horas de bajo tráfico, permite respetar a los sitios en los cuales estamos realizando extracción de información. Esto es especialmente importante cuando la anatomía del scraper es uno-a-uno.

Web scraping en node.js

Node.js es una elección obvia a la hora de implementar un web scraper, pues permite manipular el DOM en del lado del servidor y es grandioso para el manejo de *request* y eventos de I/O asíncronos.

Los módulos de node.js más útiles para realizar scraping de datos de páginas web y manipulación del DOM son los siguientes:

- **Módulo url**

  Mediante el módulo url se realiza la resolución y análisis sintáctico de la URL específica donde deseamos hacer scraping. Esto nos permite hacer uso de los

diferentes segmentos de la URL de manera mas transparente, sin tener que recurrir a funciones de manipulación de cadenas de texto. Éste, es particularmente útil cuando la URL de un enlace es relativa y debemos resolverla respecto a la URL base, para poder visitarlos en la siguiente iteración.

```
var url = require('url');
link = url.resolve('http://hdmagazine.org', '/conocenos');

console.log(link);
```

- **Módulo http**

  Este módulo permite, entre muchas otras funciones, realizar solicitudes a URL específicas y añadir un *listener* donde será entregada la respuesta a la solicitud, que ejecutará una llamada de retorno cuando se produzca el evento `response`.

```
var http = require('http');
var options = { hostname: 'hdmagazine.org' };

var body = '';
http.get(options, function(response) {
	response.on('data', function(chunk) {
		body += chunk;
	});
	response.on('end', function(){
		console.log(body);
	})
});
```

- **Módulo jquery**

  El módulo de jquery para node.js requiere instalación mediante npm (Node Package Manager), pero permite realizar un análisis sintáctico del DOM de manera muy fácil. Después de haber instalado jquery, es posible definir la variable $, que nos permitirá seleccionar elementos del DOM como en jquery. Complementando el ejemplo anterior:

```
var $ = require('jquery');
var http = require('http');
var options = { hostname: 'hdmagazine.org' };

var body = '';
http.get(options, function(response) {
	response.on('data', function(chunk) {
		body += chunk;
	}).on('end', function() {
		var title = $(body).find('title').text();
		console.log(title);
	});
});
```

Node.io

Entre las numerosas librerías escritas en node.js, que existen para realizar automatización de requerimientos de I/O, está node.io, que permite crear un objeto *job* con los siguientes métodos:

- **`input()`:** Datos de entrada o semilla a procesar por el job. Por defecto lee el STDIN interpretando los saltos de línea con \n o \r. Los datos de entrada también pueden ser: un array, un path_to_file (procesa cada una de las líneas del archivo) o path_to_dir (procesa cada una de los archivos del directorio), o un stream.
- **`output()`:** Datos de salida del job. Por defecto se presentan en el STDOUT, pero también puede especificarse la salida a un path_to_file o un stream.
- **`run()`:** Procesa los datos de entrada y emite el resultado a la salida. En la función de llamada de retorno que se define para este método, es donde encapsulamos los métodos de *scraping* de datos.[4]

Entre muchos otros métodos de node.io, se incluyen dos muy importantes para realizar scraping de datos de páginas web: `get()` y `getHtml()`, que permiten hacer extraer los datos de una URL. Específicamente, la función de llamada de retorno de `getHtml()` permite tomar el argumento $ (el cual es un objeto similar al $ de jquery) y acceder al DOM usando selectores para procesar la información relevante para nuestro job. Por ejemplo:

```
var nodeio = require('node.io');
exports.job = new nodeio.Job({
    //Se establece input en false, el job se ejecuta solo 1 vez.
    input: false,
    run: function () {
          this.getHtml('http://hdmagazine.org', function (err, $) {
                var links = [];
                //Para cada elemento <a> encontrado se obtiene el atributo href
                $('a').each(function(data) {
                      links.push(data.attribs.href);
                });
                this.emit(links);
          });
    }
});
```

A su vez, el método `getHtml` permite hacer *crawling* a través de motores de búsqueda. Por ejemplo, si deseamos conocer los diferentes subdominios de un sitio web, podemos utilizar el motor de búsqueda Google en node.io[5]:

```
var nodeio = require('node.io');

//El dominio semilla se ingresa como argumento en CLI
var domain = process.argv[3];
var subs = [''];
//Elementos de la URL de busqueda
```

4 Para conocer el resto de los métodos, visitar la documentación de la API https://github.com/chriso/node.io/wiki
5 Este ejemplo puede ser encontrado en https://github.com/eramirem/node.js/blob/master/domain-crawler.js

```
var base_url = 'https://www.google.com';
var base_path = '/m/search?';
var base_query = 'site:' + domain;
//Expresión regular que haga fetch con un dominio/subdominio
var pattern = new RegExp('(\\w+):\\/\\/([\\w-.]+' + domain + ')');
var query = '';
//Máximo número de palabras que el motor de búsqueda admite en el query
const max_words = 32;
//Longitud máxima de la URI
const max_uri = 2048;

//Tiempo de espera entre iteraciones de 5 segundos
//Tiempo de espera para la respuesta de 10 segundos
exports.job = new nodeio.Job({ wait: 5, timeout: 10 }, {
    //El job se ejecuta indefinidamente hasta que se presente una señal de exit
    input: true,
    run: function () {
           full_query = base_query + query;
           //50 resultados por página
           start_param = '&num=50&start=0';
           query_param = 'q=' + full_query;
           params = query_param + start_param;
           full_url = base_url + base_path + params;
           console.log('Query ' + full_url);

           if (subs.lenght > max_words) this.exit('Max words limit reached');
           if (full_url.length > max_uri) this.exit();

           this.getHtml(full_url, function (err, $) {
                  var new_domain = false;
                  if($('#resultStats').text == false) this.exit('No results');

                  //Se obtiene el link del header de cada resultado
                  $('h3 a').each('href', function(href) {
                         //Si el link coincide con el patrón de subdominio
                         if (href.match(pattern))
                         {
                                link = href.match(pattern)[2];
                                //Si el link no se encuentra en el arreglo existente
                                if(subs.indexOf(link) == -1)
                                {
                                       new_domain = true;
                                       console.log('New domain found: ' + link);
                                       query += '+-site:' + link;
                                       subs.push(link);
                                }
                         }
                  });
                  //Si no se encuentran subdominios nuevos en la última iteración
                  if (new_domain == false) this.exit('No new domains found');
                  this.emit();
           });
    }
});
```

Para ejecutar el script anterior, se utiliza la línea de comandos, pasando como argumentos el nombre del script y el dominio a evaluar:

```
$ node.io domain.js hdmagazine.org
```

La salida del script sería la siguiente:

```
$ node.io test.js hdmagazine.org
Query https://www.google.com/m/search?q=site:hdmagazine.org&num=50&start=0
New domain found: store.hdmagazine.org
New domain found: www.hdmagazine.org

Query https://www.google.com/m/search?q=site:hdmagazine.org+-
site:store.hdmagazine.org+-site:www.hdmagazine.org&num=50&start=0
ERROR: No results
```

Básicamente la utilidad de node.io, radica en la automatización de los jobs. Sin embargo, la lógica del *scraping* sigue siendo implementada por el usuario.

Conclusión

Los web scrapers son muy útiles para implementar en nuestros *scripts* múltiples tareas que realizamos normalmente de manera manual con el navegador, como monitorización de tarifas, tasas de cambio, acciones de bolsa, agregación de noticias, etc.

Hay que tener las precauciones del caso, como evitar afectar el desempeño del sitio que se analiza evitando ataques de denegación de servicio, no deseados. También tener en cuenta material protegido por copyright. El propósito para el cual la información obtenida mediante un web scraper es usada, debe ser autorizado por el generador de dicho contenido.

¿Cómo empiezo con JavaScript?

¿Te gustaría usar JavaScript en el desarrollo de tus aplicaciones? Conoce cómo puedes empezar con JavaScript y las ventajas que nos da al momento de "codear".

Escrito por: **Milagros Alessandra Infante Montero** (Est. Ing. Informática)

Estudiante de Ingeniería Informática. Miembro de **APESOL** (Asociación Peruana de Software Libre) y de la comunidad de software libre **Lumenhack**. Miembro del equipo de traducción al español de **GNOME**. Apasionada por el desarrollo de software, tecnología y gadgets. Defensora de tecnologías basadas en software libre y de código abierto.

Webs:
Blog: www.milale.net

Redes sociales:
Twitter / Identi.ca: **@milale**

JavaScript nació en los años '90 cuando se empezaba el desarrollo de aplicaciones web cada vez más complejas. Para conectarse a Internet, las personas utilizaban módems que otorgaban una velocidad de conexión total de 56 Kbps, que no llegaba a superar los 28.8 Kbps. Por esto, era necesario que existiese un lenguaje que al rellenar de manera incorrecta un formulario, por ejemplo, no se tuviese que esperar mucho hasta que el servidor lo procesara y volviera a mostrarlo vacío. De esta forma, JavaScrip, se ejecutaba -y aún se ejecuta- en el navegador del usuario y no en el servidor, acelerando así todo el proceso.

JavaScript es mejor denominado como ECMAScript (es el lenguaje definido por el primer estándar ECMA-262[6] creado por el comité TC39) ya que es la implementación de la empresa *Netscape* sobre este estándar.

¿Por qué JavaScript?

JavaScript es muy utilizado en el desarrollo y diseño de sitios web. Los navegadores directamente interpretan este tipo de código. Muchas veces, se lo confunde con Java pero son lenguajes diferentes y singulares. JavaScript puede ser añadido en cualquier

6 http://www.ecma-international.org/publications/standards/Ecma-262.htm

página web y ser ejecutado sin la necesidad de otro programa o *framework*. Actualmente puede funcionar en correo, chat, buscadores, calculadoras, validadores de formularios y un extenso etcétera. En algún momento perdió popularidad con la aparición de Flash pero la recobró cuando nació AJAX. Cuando se firma digitalmente un *script* y se solicita al usuario permiso para realizar esas acciones, se superan algunas limitaciones que el lenguaje presenta, por ejemplo si la ejecución de un *script* dura mucho tiempo quizás por un error, el navegador informa al usuario que demasiados recursos están siendo consumidos y da la posibilidad de detener la ejecución.

¿Cómo incluyo JavaScript en (X)HTML?

Incluir código JavaScript en (X)HTML es muy flexible. Veremos algunas formas de hacerlo:

1. **JavaScript en el mismo documento XHTML**

   Colocando las etiquetas `<script>` y `</script>` se puede incluir en cualquier parte del documento.

   > Se recomienda tener este código en la cabecera del documento dentro de las etiquetas `<head>` y `</head>`

   Dentro de esta parte del código se debe añadir el atributo `type` y según el estándar, el valor que debe agregarse es `text/javascript`. Este método se emplea en pequeños bloques de código y si se desea cambiar algo, debe hacerse en todas las partes donde éste se encuentre.

```
<html>
  <head>
    <script type="text/javascript">
      document.write("Hola lectores de H&D Magazine");
    </script>
  </head>
  <body>
    <p>Texto en HTML</p>
  </body>
</html>
```

2. **JavaScript en los elementos (X)HTML**

   Se introducen instrucciones de código dentro de etiquetas (X)HTML, accionándose los mismos mediante un evento del usuario. Pero no es muy usado ya que no es una buena práctica de programación.

```
<html>
    <head>
           <meta  content="text/html; charset=iso-8859-1" />
```

```
        <title>Ejemplo de código JavaScript en el propio documento</title>
    </head>
    <body>
        <p onclick="alert('Probando')">Texto del párrafo.</p>
    </body>
</html>
```

3. **JavaScript en un archivo externo**

   Esta se consideraría como la mejor práctica de programación ya que de esta manera el código estaría limpio, más manejable al hacer cambios y se reutilizaría al tener un archivo exclusivo para código JavaScript. Se enlazan con la etiqueta `<script>` a un documento (X)HTML y no hay límite de cantidad de archivos JavaScript a añadir. Se define el atributo `src` donde se indica la URL del archivo en JavaScript (archivo con extensión .js) a enlazar.

> Se puede enlazar un único archivo en cada etiqueta <script>, pero en una misma página se pueden incluir tantas etiquetas <script> como se necesiten.

```
// Archivo ejemplo.js

for (a=1; a<=6; a++){
    document.write('<h' + a + '>Cabecera de nivel' + a);
    document.write('</h' + a + '>');
}


<!-- documento HTML -->
<html>
    <head>
        <title>Ejemplo de JavaScript</title>
        <script type="text/javascript" src="/js/ejemplo.js"></script>
    </head>
    <body>
        <p>Texto del archivo.</p>
    </body>
</html>
```

Palabras reservadas para JavaScript

```
break, case, catch, continue, default, delete, do, else, finally, for, function,
if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void,
while, with
```

Ahora veamos, la sintaxis...

Existen muchos aspectos importantes a tomar en cuenta:

- **Distingue entre mayúsculas y minúsculas:** si se utilizara indistintamente, el *script* no funcionará.
- **Los espacios en blanco (tabulaciones y nuevas líneas) son ignorados:** Para ordenar el código se puede tabular, crear nuevas líneas para una misma instrucción, añadir espacios, etc. ya que el intérprete ignorará cualquier espacio en blanco, esperando el final de la instrucción.
- **Las instrucciones finalizan en punto y coma:** Aunque no sea obligatorio para todas las sentencias, sí lo es para una gran parte de las instrucciones. Utilizar un punto y coma para finalizar cada instrucción, mantendrá el orden y es considerado una buena práctica de programación.
- **No se define tipo de variable:** Una variable puede almacenar diferentes tipos al ejecutar un script ya que es de tipado dinámico.
- **Uso de comentarios:** Se pueden añadir para documentar el código como una buena práctica ya que al igual que en otros lenguajes, éstos no se mostrarán en pantalla.

```
//comentarios de una sola línea
alert("Hola HD Magazine");

/*De esta manera podemos añadir comentarios
de varias líneas cuando se necesite detallar
mucha información*/
alert("hola HD Magazine");
```

Variables

Las variables están destinadas a almacenar datos. Para declarar una variable, se antecede a ésta, la palabra reservada `var`. JavaScript también permite el uso de variables sin previa declaración. Cualquier variable definida en el ámbito del documento, será interpretada como variable global, pudiendo accederse a ella dentro de cualquier función. Sin embargo, una variable definida dentro de una función, no será accesible fuera de ella. El símbolo utilizado para la asignación es =.

> Es muy recomendable para evitar futuros errores y como una buena práctica, declarar todas las variables que se vayan a usar.

El nombre que se le da a la variable es el identificador y se debe tener sumo cuidado en que el primer carácter no puede ser un número y el resto de la palabra puede estar formado por letras, números y con los símbolos $ y _.

Tipos de datos de las variables

Las variables, en JavaScript pueden ser de diversos tipo:

Numéricas:

El valor se asigna directamente si es *int* (entero) o si es *float* (decimal) donde se coloca un punto en vez de la coma.

```
var foo = 90;
var bar = 75.8;
```

Cadenas de texto:

Se utiliza comillas simples `'texto'` o dobles `"texto"` para delimitar su inicio y fin.

```
var foo = 'Hola Mundo!';
```

Booleanas:

Toma valores `true` o `false` (verdadero o falso).

```
var foo = true;
```

Arrays: Se puede colocar una colección de datos que pueden ser o no del mismo tipo:

```
var meses = ['Enero', 'Marzo', 'Mayo', 'Julio', 'Setiembre', 'Noviembre'];
var datos_persona = ['Juan', 'Pérez', 25, 'Perú', false];
```

Para acceder a cada uno de sus elementos, solo se debe indicar la posición y debemos recordar que la numeración empieza en 0 hasta N.

```
var edad_de_juan = datos_persona[2];
// salida: 25

var casado = datos_persona[4];
// salida: false
```

Para incrementar una variable se utiliza `++` y para decrementar se utiliza `--`

Si se utiliza como prefijo (delante de la variable) el valor se incrementa antes de la operación, pero, si se utiliza como sufijo (después de la variable) el valor se incrementa luego de ejecutar la operación.

```
var foo = 25;
document.write(foo++);
// salida: 25
document.write(foo);
```

```
// salida: 26

var bar = 25;
document.write(++bar);
// salida: 26
```

Es importante recordar que mientras aprendes a programar en cualquier lenguaje, un *cheat sheet* siempre será de mucha ayuda al desarrollar[7].

Sencillos «snippets» en JavaScript, para NO programadores

Cada elemento (etiqueta) de un documento HTML, es tratado como un objeto en JavaScript. JavaScript puede identificar estos objetos, leer sus propiedades (atributos) y también modificarlos. La forma más simple de acceder, desde JavaScript a un elemento del HTML, es a través de su id. Todo elemento en HTML (tag) puede tener su atributo id. La id de un elemento debe ser única y no puede repetirse. Por ejemplo:

```
<a href="http://www.google.com" id="LinkGoogle">Google</a>
<input type="text" name="nombre_apellido" id="NombreApellido"/>
```

Desde JavaScript, se accederá a los elementos anteriores, identificándolos como objetos, mediante su ID:

```
document.getElementById('idDelElemento');
```

Por ejemplo:

```
document.getElementById('LinkGoogle');
```

Las estructuras de control, se agrupan entre dos llaves y las funciones se definen con la palabra `function`. Por ejemplo:

```
    function mi_funcion() {
        var a = 10;
        var b = 15;
        var suma = a + b;
        return suma;
    }
```

7 http://www.addedbytes.com/cheat-sheets/javascript-cheat-sheet/

Y en JavaScript, las funciones también pueden recibir parámetros:

```
    function mi_funcion(a, b) {
        var suma = a + b;
        return suma;
    }
```

Desde cualquier elemento HTML se puede llamar a una función o instrucción JavaScript (como se comentó anteriormente). Para ello, las llamadas se suelen realizar mediante un evento concreto del usuario. Entre los eventos más habituales, se pueden encontrar:

onclick

E usuario pulsa el botón izquierdo del ratón sobre el elemento. Es un evento disponible desde la mayoría de los elementos HTML. Por ejemplo:

```
<p onclick="alert('Esto es un párrafo');">Clic aquí</p>

<!-- en un formulario -->
<input type="radio" name="opcion" value="SI"
    onclick="alert('Elegiste la opción SI');"/>
```

onkeydown / onkeyup

El usuario pulsa una tecla / el usuario levanta “el dedo” de una tecla :) Mayormente se utiliza en campos de formulario. Por ejemplo:

```
<input type="text" name="edad" onkeyup="alert('escribiste algo');"/>
```

onchange

En un campo de formulario, por ejemplo un `select`, se refiere a cuando el usuario, cambia la opción seleccionada. Por ejemplo:

```
<select name="pais" onchange="alert('Cambiaste de opción');">
    <option>Argentina</option>
    <option>España</option>
    <option>Perú</option>
</select>
```

Un uso frecuente de JavaScript, es por ejemplo, modificar los estilos (CSS) de diversos elementos, tras un evento determinado. Por ejemplo, la siguiente instrucción, oculta el elemento cuya id es "parrafo2":

```
document.getElementById('parrafo2').style.display = 'none';
```

Y la siguiente, hace visible el mismo elemento:

```
document.getElementById('parrafo2').style.display = 'block';
```

Y esta otra, cambia el color de fuente:

```
document.getElementById('parrafo2').style.color = '#f60';
```

Si se coloca esta instrucción en una función, luego se la pueda llamar tras cualquier evento. Por ejemplo:

```
// archivo .js
function cambiar_color() {
    document.getElementById('parrafo2').style.color = '#f60';
}

<!-- documento HTML -->
<p id="p1" onclick="cambiar_color();">Click aquí para cambiar el color del
    párrafo 2</p>

<p id="parrafo2">Este párrafo cambiará de color</p>
```

Podrías modificar aún más la función anterior y hacer que la ID del elemento se pase por parámetro. Por ejemplo:

```
function cambiar_color(id) {
    document.getElementById(id).style.color = '#f60';
}
```

Entonces, usar la misma función para cambiar el color de fuente de diversos elementos:

```
    <b id="b1" onclick="cambiar_color('b1');">Pulsar y cambiar mi color</b><br/>
    <b id="b2" onclick="cambiar_color('b2');">Pulsar y cambiar mi color</b><br/>
    <b id="b3" onclick="cambiar_color('b3');">Pulsar y cambiar mi color</b><br/>
```

Cuando se desea escribir, con JavaScript, un texto en pantalla, se puede hacer de forma "más prolija", utilizando las siguientes instrucciones:

```
function escribir(id) {
    var texto = 'Nuevo Texto';

    // para la mayoría de los navegadores
    document.getElementById(id).innerText = texto;
```

```
    // para Firefox
    document.getElementById(id).contentText = texto;
}
```

Por ejemplo:

```
    <p id="p1"></p>
    <input type="button" value="Escribir" onclick="escribir('p1');"/>
```

Y si se desea, además de texto, escribir código HTML, se utiliza `innerHTML`. Por ejemplo:

```
// archivo .js
function escribir_html(id) {
    var codigo = "<b>Hola Mundo!</b>";
    document.getElementById(id).innerHTML = codigo;
}

<!-- documento HTML -->
<div id="layer1"></div>
<input type="button" value="Escribir HTML" onclick="escribir_html('layer1');"/>
```

En JavaScript se pueden utilizar condicionales `if/else`. Por ejemplo:

```
// archivo .js
function mostrar_u_ocultar() {
    // si boton1 está checkeado
    if(document.getElementById('boton1').checked == true) {
        // oculta el campo texto1
        document.getElementById('texto1').style.display = 'none';
        // muestra el campo texto2
        document.getElementById('texto2').style.display = 'block';
    // si no está checkeado
    } else {
        // muestra el campo texto1
        document.getElementById('texto1').style.display = 'block';
        // oculta el campo texto2
        document.getElementById('texto2').style.display = 'none';
    }
}
<form>
    <input type="radio" name="opcion" id="boton1" value="1"
        onclick="mostrar_u_ocultar()"/> 1<br/>
    <input type="radio" name="opcion" id="boton2" value="2"
        onclick="mostrar_u_ocultar()"/> 2<br/>

    Texto 1: <input type="text" name="texto1" id="texto1"/><br/>
    Texto 2: <input type="text" name="texto2" id="texto2"/><br/>
</form>
```

Conociendo a DOM: Parte I

Para poder desarrollar aplicaciones y sitios web con HTML5 es importante saber acceder y actualizar el contenido, la estructura y el estilo de documentos.

Escrito por: **Sergio Infante Montero** (Ingeniero de Software)

Ingeniero Informático con estudios de **Master de Dirección Estratégica en TI**. Ingeniero de software en **Taller Technologies,** activista, contribuidor y consultor de proyectos **FLOSS**, miembro de **APESOL** y escritor de artículos y libros técnicos de programación.

Perfiles:
http://about.me/neosergio
Twitter: **@neosergio**

El Modelo de Objetos del Documento (Document Object Model) o también conocido como DOM (por sus siglas en inglés) es una plataforma mixta y una interfaz de lenguaje neutral que permite a los *scripts* y programas acceder al contenido, estructura y estilo de documentos buscando interactuar con ellos. El resultado de esta interacción se incorpora nuevamente en la página presentada. En palabras simples, **es una manera de acceder a los objetos en pantalla**.

El World Wide Web Consortium (W3C) está a cargo de la especificación y del esfuerzo de estandarización de DOM[8], como es de suponer por obvias razones, el objetivo de W3C es mantener solo un estándar.

Con el DOM se puede tener control de casi todo elemento de una página; crear objetos, alterar su contenido, cambiar la forma de la página y mucho más. Si se desea tener mayor referencia técnica se puede consultar la documentación del DOM Core[9] y del DOM HTML[10].

8 http://www.w3.org/DOM/
9 http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/level-one-core.html
10 http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/level-one-html.html

Estructura Jerárquica

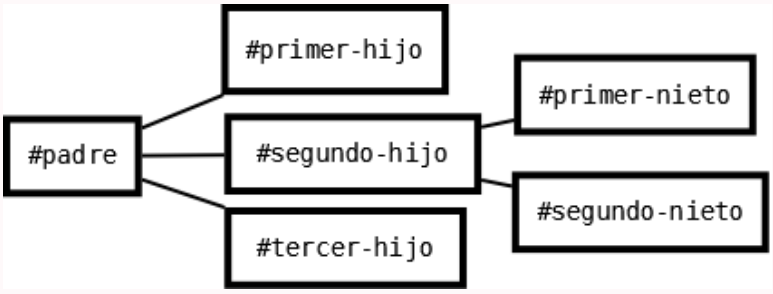
La estructura jerárquica del DOM es similar a la de un árbol familiar; entenderla así es fundamental. De las palabras que forman la sigla, el *document*, se refiere a la página actual y su contenido. Dentro de la estructura jerárquica, el *document* representa la parte superior de la estructura y todo lo demás forma parte de el:

> *Por ejemplo la etiqueta* ***<html>*** *forma parte del document y la etiqueta* ***<body>*** *forma parte del* ***<html>*** *-al igual que* ***<head>****-, por lo cual podemos decir que también forman parte del document e imaginarlos como si fuera un árbol familiar.*

Es una jerarquía de objetos que se les conoce como nodos. Hay diversos tipos de nodos y un nodo puede tener varios de estos:

Tipo de objeto	Interpretación
`parentNode`	Nodo tipo padre
`childNodes`	Lista de nodos hijos
`firstChild`	Nodo del primer hijo
`lastChild`	Nodo del último hijo
`previousSibling`	Nodo hermano anterior al actual
`nextSibling`	Nodo hermano posterior al actual

Aquí hay un ejemplo que permitirá entender mejor:



Para poder usar el gráfico anterior como ejemplo vamos a emplear JavaScript y el método `getElementById`[11].

11 https://developer.mozilla.org/en-US/docs/DOM/document.getElementById

```
elemento = document.getElementById(id);
```

Si aún no dominas JavaScript, te recomiendo leer el artículo "¿Cómo comienzo con JavaScript?" en la página 17 de esta edición.

Considerar que el id vendría a ser la manera única de ubicar el elemento dentro del documento y por esta razón, no deben existir id duplicadas en un mismo documento.

Algunas maneras de acceder a los elementos

Podemos entonces usar esta estructura jerárquica para acceder a los elementos de un documento.

Usando la imagen anterior podemos tener entonces, algunos ejemplos:

```
// Para acceder al segundo hijo
segundoHijo = document.getElementById('segundo-hijo');

// Para acceder al padre del segundo hijo
padre = document.getElementById('segundo-hijo').parentNode;

// Para acceder a los hijos del segundo hijo
nietos = document.getElementById('segundo-hijo').childNodes;

// Para acceder al primer hijo a través del padre
primerHijo = document.getElementById('padre').firstChild;

// Para acceder al tercer hijo (último) a través del padre
ultimoHijo = document.getElementById('padre').lastChild;

// Para acceder a los hermanos del segundo hijo
hermanoAnterior = document.getElementById('segundo-hijo').previousSibling;
hermanoPosterior = document.getElementById('segundo-hijo').nextSibling;

// Otra manera de acceder a los nietos
primerNieto = document.getElementById('segundo-hijo').childNodes.item(0);
```

Ahora veamos un ejemplo un poco más completo y enfocado a HTML. Tenemos la siguiente estructura HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>
```

```
        <nav>
            <ul id='lista'>
                <li id='elemento1'>
                    <a id='enlace1'>Primer enlace</a>
                </li>
                <li id='elemento2'>
                    <a id='enlace2'>Segundo enlace</a>
                </li>
                <li id='elemento3'>
                    <a id='enlace3'>Tercer enlace</a>
                </li>
            </ul>
        </nav>
    </body>
</html>
```

Para acceder a los elementos, ahora vamos a usar un método más: `getElementsByTagName`[12] (notar que está en plural, a diferencia del anterior que está en singular).

```
// Algunas formas de acceder a la lista
document.getElementById('lista');
document.getElementsByTagName('ul').item(0);
document.getElementsByTagName('li').item(0).parentNode;
document.getElementsByTagName('li').item(1).parentNode;
document.getElementsByTagName('a').item(0).parentNode.parentNode;

// Algunas formas de acceder a la sección de navegación
document.getElementById('lista').parentNode;
document.getElementsByTagName('body').item(0).childNodes.item(0);
document.body.childNodes.item(0);
```

Un método más que puedes investigar es `document.body`[13]

Algunas propiedades, métodos y atributos de los nodos

Ahora que ya sabemos que podemos acceder a los elementos de varias formas, es importante aprender que propiedades, métodos y atributos podemos usar:

nodeName

Nos retorna el nombre de un nodo:

```
document.getElementById('enlace1').nodeName;
```

12 https://developer.mozilla.org/en/docs/DOM/document.getElementsByTagName
13 https://developer.mozilla.org/en-US/docs/DOM/document.body

tagName

Nos retorna el nombre de la etiqueta del elemento:

```
document.getElementById('enlace1').tagName;
```

nodeType

Devuelve 1, 2 o 3. Estos números representan los tipos de nodos. Para el número 1 es elemento; 2 es atributo y 3 es texto:

```
document.getElementById('elemento2').nodeType;
```

hasChildNodes()

Devuelve verdadero o falso al consultar si el elemento tiene nodos hijos:

```
document.getElementById('lista').hasChildNodes();
```

nodeValue

Devuelve el valor del nodo:

```
document.getElementBy('enlace2').firstChild.nodeValue;
```

setAttribute(atributo, valor)

Configura un atributo con el valor dado:

```
document.getElementById('lista').setAttribute('display', 'inline');
```

getAttribute(atributo)

Recibe el atributo como argumento y devuelve su valor:

```
document.getElementById('lista').getAttribute('display');
```

removeAttribute(atributo)

Remueve el atributo:

```
document.getElementById('lista').removeAttribute('display');
```

Es momento de practicar con todos estos elementos y lo visto en este artículo. Un ejemplo de código para practicar sería el siguiente:

```
<!DOCTYPE html>
<html>
    <head>
           <title>Ejemplo</title>
        <script type="text/javascript">
           function probando(){
                  variable = document.getElementById('lista').nodeName;
                  alert(variable);
            };
        </script>
    </head>
    <body>
           <nav>
                  <ul id='lista'>
                         <li id='elemento1'>
                                <a id='enlace1'>Primer enlace</a>
                         </li>
                         <li id='elemento2'>
                                <a id='enlace2'>Segundo enlace</a>
                         </li>
                         <li id='elemento3'>
                                <a id='enlace3'>Tercer enlace</a>
                         </li>
                  </ul>
           </nav>
        <h1 onClick='probando()'>Clic aquí para probar. Sí, dale clic aunque
          no parezca un enlace</h1>
    </body>
</html>
```

Para probar código se puede usar http://jsfiddle.net, http://jsbin.com o incluso http://dabblet.com/. Puedes intentar también reemplazar **alert** por **console.log** y usar la consola de depuración de javascript, presente en varios navegadores web.

Manual de MVC: (3) Los objetos View

En el capítulo anterior, vimos como identificar los diferentes tipos de sustituciones que desde las vistas de los modelos, en MVC, pueden realizarse. En esta entrega veremos como crear los objetos View de cada modelo, los cuáles implementarán todo lo aprendido en el capítulo anterior.

Escrito por: **Eugenia Bahit** (Arquitecta GLAMP & Agile Coach)

Eugenia es **Arquitecta de Software**, **docente** instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y eXtreme Programming. Miembro de la **Free Software Foundation** e integrante del equipo de **Debian Hackers**.

Webs:
Cursos de programación a Distancia: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

En el capítulo anterior, estuvimos viendo ejemplos de los diferentes algoritmos que pueden utilizarse tanto en Python como en PHP, para realizar sustituciones estáticas y dinámicas (*render*) en los archivos HTML.

Dichos algoritmos, formarán parte de los métodos de cada una de las vistas, pudiendo además, crearse objetos `View` a nivel del `core`, para ser reutilizados en las vistas de cada modelo.

> *En MVC, cada modelo debe contar con una vista (objeto ModelView)*

En principio, debemos saber que para cada modelo debe haber un objeto vista (objeto View). Sin embargo, no todos los recursos, deberán tener un método en la vista (pero sí, en el controlador del modelo, que veremos más adelante).

Cuando en MVC se poseen las funcionalidades propias de un ABM, como agregar, modificar y eliminar un objeto, las vistas, deberán tener al menos, dos métodos: uno para mostrar el formulario de alta y otro, para mostrar el formulario de edición. En los

ABM, también son frecuentes otros dos métodos: uno para mostrar el listado de una colección de objetos determinada y otro, para visualizar un objeto puntual. Por supuesto, la cantidad de métodos de una vista, depende solo y exclusivamente de los requerimientos gráficos de cada aplicación. No obstante, el objetivo de esta entrega, es ver como crear los objetos View. Luego, la cantidad de métodos a desarrollar, dependerá del lector y de los requerimientos de su GUI.

Los métodos de la vista, serán invocados por el controlador del modelo. Éste, será quien entregue los datos a la vista, mientras que la última, será la encargada de realizar las sustituciones pertinentes e imprimir el resultado de las mismas en pantalla. Básicamente, la secuencia podría describirse como la siguiente:

1. El usuario solicita un recurso a través del navegador
2. El `FrontController` (descrito en la edición #1), analiza la petición del usuario e instancia al controlador correspondiente (veremos controladores más adelante)
3. El constructor de cada controlador, es quien realiza una llamada de retorno a un método propio el cual se encargará de:
    - Instanciar al modelo correspondiente (aunque generalmente, esta instancia se realiza desde el constructor en la mayoría de los casos)
    - Realizar las modificaciones necesarias sobre el modelo (modificar propiedades, llamar a los métodos necesarios, etc.)
    - Entregar los datos retornados por el modelo a un método de la vista.
4. La vista, traerá las plantillas necesarias, para sustituir los datos que le han sido entregados por el controlador. Finalmente, imprimirá en pantalla, el resultado de dicha sustitución.

> *Antes de continuar, si aún no lo has hecho,* ***te recomiendo leer la segunda entrega del manual de MVC***[14] *y practicar con los ejemplos allí descritos, a fin de poder comprender mejor, todo lo expuesto en este artículo*

Cómo bien se comentó, debe existir una vista (objeto View) por cada modelo. El nombre de cada vista, generalmente, será el nombre del modelo, seguido de la palabra View. Por ejemplo, dados los modelos: `Vidrio`, `Marco` y `Ventana`, tendremos 3 vistas: `VidrioView`, `MarcoView` y `VentanaView`:

```
# en PHP
class VidrioView {
}
```

14 http://www.hdmagazine.org/?magazine=HackersAndDevelopers&num=2

```
# en Python
class VidrioView(object):
    pass
```

Suponiendo que para nuestro modelo `Vidrio` hayamos definido un recurso `agregar` (que deberá ser un método del controlador, como veremos más adelante), **en principio, debemos contar con la GUI** para este recurso (aquí no incluiremos aún la plantilla general, sino solo el contenido relativo a este recurso. La plantilla general se incluirá al final de este artículo):

```
<h3>Agregar un nuevo vidrio</h3>
<form method="POST" action="/mimodulo/vidrio/guardar">
<!-- notar que el nombre de los campos, debe coincidir con el nombre de las
propiedades del objeto, siempre que esto sea posible-->
    <label for="grosor">Grosor:</label><br/>
    <input type="text" name="grosor" id="grosor" size="3"/> mm<br/><br/>
    <label for="sup">Superficie:</label><br/>
    <input type="text" name="superficie" id="sup"/><br/><br/>
    <label for="color">Color:</label><br/>
    <input type="text" name="color" id="color"/><br/><br/>
    <input type="submit" value="Guardar"/>
</form>
```

Como podemos observar, la GUI siempre debe ser el primer paso en el proceso de desarrollo de las vistas.

En este ejemplo en particular, nos encontramos con que la GUI, no requiere de ninguna sustitución (ni estática ni dinámica). Entonces, lo único que necesitaremos tener en la lógica de nuestra vista, es un método que traiga dicha GUI y la muestre en pantalla. Este método, no necesariamente debe llamarse `agregar()`. Podría tener un nombre más descriptivo como por ejemplo, `mostrar_form_alta()`:

```
# PHP: Archivo /myapp/modulo/views/vidrio.php
class VidrioView {

    function __construct() {
    }

    function mostrar_form_alta() {
        $plantilla = file_get_contents("/ruta/a/agregar_vidrio.html");
        print $plantilla;
    }

}
```

```
# Python: Archivo /myapp/modulo/views/vidrio.py
class VidrioView(object):

    def __init__(self):
        pass
```

```
    def mostrar_form_alta(self):
        with open("/ruta/a/agregar_vidrio.html", "r") as archivo:
            plantilla = archivo.read()
        print plantilla
```

Cuando el formulario sea enviado, será solicitado el recurso `guardar` (ver atributo "`action`" del formulario HTML). Este recurso, será procesado por el controlador pero ¿qué haremos finalmente con el usuario? El controlador, podrá decidir que sería una buena idea, mostrarle el objeto creado, al usuario. En ese caso, deberá existir un recurso "`ver`" que también será manejado por el controlador (al igual que todos los recursos). El recurso `ver` será un método del controlador. Éste, se encargará de recuperar el objeto `Vidrio` recién creado y entregárselo a la vista para que haga lo suyo. Entonces ¿qué debemos tener primero? La GUI, igual que siempre:

```
<!-- GUI para PHP -->
<h3>Vidrio ID {vidrio_id}</h3>
<b>Grosor:</b> {grosor} mm<br/>
<b>Superficie:</b> {superficie}<br/>
<b>Color:</b> {color}
```

```
<!-- GUI para Python -->
<h3>Vidrio ID $vidrio_id</h3>
<b>Grosor:</b> $grosor mm<br/>
<b>Superficie:</b> $superficie<br/>
<b>Color:</b> $color
```

En este caso, la vista deberá contar con un método, que se encargue de realizar una **sustitución estática**. Para esto, el controlador, le deberá pasar un objeto `Vidrio` como parámetro. Ampliemos el ejemplo anterior:

```
# PHP: Archivo /myapp/modulo/views/vidrio.php
class VidrioView {

    function __construct() {
    }

    function mostrar_form_alta() {
        $plantilla = file_get_contents("/ruta/a/agregar_vidrio.html");
        print $plantilla;
    }

    function mostrar_objeto($objeto_vidrio) {
        # Traigo la plantilla
        $plantilla = file_get_contents("/ruta/a/ver_vidrio.html");

        # Creo el diccionario
        settype($objeto_vidrio, 'array');
        foreach($objeto_vidrio as $clave=>$valor) {
            $objeto_vidrio["{{$clave}}"] = $valor;
```

```
            unset($objeto_vidrio[$clave]);
        }

        # Realizo la sustitución
        $render = str_replace(array_keys($objeto_vidrio),
            array_values($objeto_vidrio), $plantilla);

        # Imprimo el resultado en pantalla
        print $render;
    }

}
```

```
# Python: Archivo /myapp/modulo/views/vidrio.py
from string import Template


class VidrioView(object):

    def __init__(self):
        pass

    def mostrar_form_alta(self):
        with open("/ruta/a/agregar_vidrio.html", "r") as archivo:
            plantilla = archivo.read()
        print plantilla

    def mostrar_objeto(self, objeto_vidrio):
        # Traigo la plantilla
        with open("/ruta/a/ver_vidrio.html", "r") as archivo:
            plantilla = archivo.read()

        # Creo el diccionario
        diccionario = vars(objeto_vidrio)

        # Realizo la sustitución
        render = Template(plantilla).safe_substitute(diccionario)

        # Retorno el resultado
        # para que application lo imprima en pantalla
        return render
```

Luego, si el resultado obtenido, se desea imprimir dentro de una plantilla general (plantilla de base), solo será cuestión de crear un método `show()` en una vista a nivel del `core`. La plantilla general, deberá contar con al menos un parámetro de sustitución para el contenido y será la vista de cada modelo, quien le pase a `show()` el valor de sustitución del mismo para finalmente, imprimir en pantalla el resultado retornado por `show()`. Aquí, una vez más, debemos contar previamente con la GUI:

```
<!-- PLANTILLA HTML PARA PHP -->
<!doctype html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="stylesheet" type="text/css" href="/static/css/style.css"/>
```

```
    <link rel="icon" href="/static/img/favicon.png" type="image/png"/>
    <title>{APP_TITLE}</title>
</head>
<body>
    <header>
        <h1>{APP_TITLE}: {MODULE_TITLE}</h1>
        <nav>
            <a href="/">{APP_TITLE}</a> &gt;
            <a href="/{MODULO}">{MODULE_TITLE}</a> &gt;
            <a href="/{MODULO}/{MODELO}">{MODEL_TITLE}</a> &gt;
            <b>{RESOURCE_TITLE}</b>
        </nav>
    </header>
    <section>
        <!-- aquí irá el contenido sustituido por la vista -->
        {CONTENIDO}
    </section>
    <footer>
        &copy; 2013 HDMagazine.org - GPL v3.0
    </footer>
</body>
</html>
```

```
<!-- PLANTILLA HTML PARA PYTHON -->
<!doctype html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="stylesheet" type="text/css" href="/static/css/style.css"/>
    <link rel="icon" href="/static/img/favicon.png" type="image/png"/>
    <title>$APP_TITLE</title>
</head>
<body>
    <header>
        <h1>$APP_TITLE: $MODULE_TITLE</h1>
        <nav>
            <a href="/">$APP_TITLE</a> &gt;
            <a href="/$MODULO">$MODULE_TITLE</a> &gt;
            <a href="/$MODULO/$MODELO">$MODEL_TITLE</a> &gt;
            <b>$RESOURCE_TITLE</b>
        </nav>
    </header>
    <section>
        <!-- aquí irá el contenido sustituido por la vista -->
        $CONTENIDO
    </section>
    <footer>
        &copy; 2013 HDMagazine.org - GPL v3.0
    </footer>
</body>
</html>
```

A nivel del core, se podrá tener una vista para la sustitución de la plantilla general. La misma, podrá realizarse mediante la llamada estática a un método de clase o a una función (fuera del contexto de una clase). Aquí, lo haremos en el contexto de una clase. Crear una clase con un método estático, es una buena alternativa para centralizar cualquier otro método relacionado con las vistas, directamente disponible desde el núcleo de la aplicación.

Por favor, notar que en los siguientes ejemplos, la constante `APP_TITLE` *se supone definida en un archivo* `settings`

```php
# Archivo: /myapp/core/view.php
class CoreView {

    public static function show($modulo, $modelo, $recurso, $render) {
        $plantilla = file_get_contents("/ruta/a/template.html");
        $diccionario = array(
            "{APP_TITLE}" => APP_TITLE,
            "{MODULE_TITLE}" => ucwords($modulo),
            "{MODULO}" => $modulo,
            "{MODELO}" => $modelo,
            "{MODEL_TITLE}" => ucwords($modelo),
            "{RESOURCE_TITLE}" => ucwords("$recurso $modelo"),
            "{CONTENIDO}" => $render
        );

        print str_replace(array_keys($diccionario), array_values($diccionario),
            $plantilla);
    }

}
```

```python
# Archivo: /myapp/core/view.py
class CoreView(object):

    def __init__(cls):
        pass

    def show(cls, modulo, modelo, recurso, render) {
        with open("/ruta/a/template.html", "r") as archivo:
            plantilla = archivo.read()

        diccionario = dict(
            APP_TITLE=APP_TITLE,
            MODULE_TITLE=modulo.title(),
            MODULO=modulo,
            MODELO=modelo,
            MODEL_TITLE=modelo.title(),
            RESOURCE_TITLE="%s %s" % (recurso.title(), modelo.title()),
            CONTENIDO=render
        )

        return Template(plantilla).safe_substitute(diccionario)
```

Luego, solo será necesario que cada una de las vistas, en vez de imprimir el resultado de la sustitución en pantalla, imprima el resultado de la llamada estática al método `show()` de `CoreView`:

```php
# PHP: Modificación del archivo /myapp/modulo/views/vidrio.php
    function mostrar($objeto_vidrio) {
        $plantilla = file_get_contents("/ruta/a/ver_vidrio.html");
        settype($objeto_vidrio, 'array');
```

```
        foreach($objeto_vidrio as $clave=>$valor) {
            $objeto_vidrio["{{$clave}}"] = $valor;
            unset($objeto_vidrio[$clave]);
        }
        $render = str_replace(array_keys($objeto_vidrio),
            array_values($objeto_vidrio), $plantilla);

        # Se imprime el resultado de la llamada estática al método show()
        print CoreView::show('modulo', 'vidrio', 'ver detalles de', $render);
    }
```

```
# Python: Modificación del archivo /myapp/modulo/views/vidrio.py
    def mostrar(self, objeto_vidrio):
        with open("/ruta/a/ver_vidrio.html", "r") as archivo:
            plantilla = archivo.read()
        diccionario = vars(objeto_vidrio)
        render = Template(plantilla).safe_substitute(diccionario)

        # Retorno el resultado de la llamada estática al método show()
        return CoreView().show('modulo', 'vidrio', 'ver detalles de', render)
```

> *Siempre digo a mis alumnos, que "el arte de las vistas en MVC, consiste en* ***lograr convertir lo que se tiene, en lo que se necesita"***

Generalmente, siempre se tendrá un objeto y lo que se necesitará, será un diccionario formado por pares de clave-valor, donde los valores, no sean del tipo colección (es decir, sean de un tipo de datos simple). **Convertir un objeto en un diccionario, es la base de la lógica de las vistas en MVC.**

En la ***próxima entrega****, nos enfocaremos en los* ***controladores*** *de los modelos, para ir finalizando nuestro Manual de MVC en Python y PHP.*

Mis primeros pasos con MongoDB

En la edición pasada se mencionaron temas referentes a NoSQL, como fue una breve introducción, servicios en la nube, la estructura interna, entre otros; en este artículo hablaré acerca de MongoDB y como puedes utilizarlo.

Escrito por: **Yecely Díaz** (M.I.A. & Desarrolladora Web)

Yecely es **Maestra en Inteligencia Artificial** y **Desarrolladora Web**. Aficionada de la Tecnología, Videojuegos y la Web. Ansiosa de aprender, contribuir y programar todas sus ideas. Ama lo que hace y el código es su vida.

Webs:
Blog: http://silvercorp.wordpress.com

Redes sociales:
Twitter: **@silvercorp**

MongoDB es un sistema de base de datos NoSQL orientado a documentos y sin estructura fija. Esto significa que cada entrada puede tener un esquema de datos diferente, con atributos que no necesariamente se repiten entre ellos. Estos datos son guardados en documentos tipo JSON (BSON en MongoDB). Es multi-plataforma y además cuenta con licencia GNU AGPL 3.0, es decir, es Software Libre.

Antes de iniciar la práctica de este artículo es necesario, conozcas las definiciones siguientes:

- **Documento.** Cada registro o conjunto de datos, es el equivalente a una fila en BD relacional.
- **Colección.** Es donde se agrupan los documentos, lo que conocemos como tablas en una BD relacional.

Para saber más sobre la estructura NoSQL y la terminología empleada, te recomiendo leer el artículo “Estructura interno, código y contexto” en la página 33, de la edición “Champagne”[15] de Hackers & Developers Magazine.

15 http://www.hdmagazine.org/?magazine=HackersAndDevelopers&num=2

Instalación de MongoDB en GNU/Linux

En **distribuciones basadas en Debian**, puede instalarse MongoDB y todas sus dependencias, mediante `apt-get`:

```
apt-get install mongodb
```

Para iniciar/parar el servicio:

```
service mongodb start
service mongodb stop
```

En **ArchLinux**, se puede instalar mediante pacman:

```
pacman -S mongodb
```

Para instalaciones en otras distribuciones como **Fedora y/o CentOS**, previamente se deberá agregar el repositorio 10gen a YUM, como se indica en el manual oficial:

http://docs.mongodb.org/manual/tutorial/install-mongodb-on-red-hat-centos-or-fedora-linux/

Para iniciar/parar el servicio:

```
service mongod start
service mongod stop
```

Para ingresar en el Shell interactivo de MongoDB:

```
mongo
```

Te desplegará un mensaje similar a:

```
MongoDB shell version: 2.0.4
connecting to: test
>
```

Esto nos indicia que ha entrado a la BD denominada “test”, que MongoDB la tiene por omisión en su instalación. Es momento de crear la nuestra, a la cual llamaremos **hd**.

```
# Crear o seleccionar la colección hd
```

```
> use hd

# Comando para mostrar colecciones
> show collections
# No desplegará nada pues aun no creamos ninguna colección
```

Insertar documentos

Crearemos nuestra primera colección que llamaremos editoras y añadiremos documentos a la misma. Recuerda que la ventaja es que no tienen una estructura fija:

```
# Insertando documentos
# db.COLECCIÓN.insert({})
> db.editoras.insert({ nombre:"Yesi", paterno:"Díaz", nick:"@silvercorp" })

# Nuevo documento con estructura diferente
> db.editoras.insert({ nombre:"Eugenia", paterno:"Bahit", nick:"@eugeniabahit",
pais:"Argentina" })

# Nuevo documento con estructura diferente
> db.editoras.insert({ nombre:"Celia", nick:"@RTFMCelia", pais:"Argentina" })
```

> *Hazlo todo tan simple como sea posible, pero no más simple"*
> *Albert Einstein*

Consultas

Tenemos 3 documentos en nuestra colección **hd**. A continuación te explicaré como realizar consultas:

```
# Buscar todos los documentos
# db.COLECCIÓN.find()
> db.editoras.find()
```

Retornará:

```
{ "_id" : ObjectId("50fc84bcb6a01eefce9b7925"), "nombre" : "Yesi", "paterno" :
"Díaz", "nick" : "@silvercorp" }
{ "_id" : ObjectId("50fc84c6b6a01eefce9b7926"), "nombre" : "Eugenia", "paterno" :
"Bahit", "nick" : "@eugeniabahit", "pais" : "Argentina" }
{ "_id" : ObjectId("50fc84ceb6a01eefce9b7927"), "nombre" : "Celia", "nick" :
"@RTFMCelia", "pais" : "Argentina" }
```

Mientras que:

```
# Mostrar únicamente un documento
# db.COLECCIÓN.findOne()
> db.editoras.findOne()
```

Retornará:

```
{
    "_id" : ObjectId("50fc84bcb6a01eefce9b7925"),
    "nombre" : "Yesi",
    "paterno" : "Díaz",
    "nick" : "@silvercorp"
}
```

Búsqueda por campo:

```
# Desplegar aquellos documentos donde la editora sea de Argentina
# db.COLECCIÓN.find({campo:"Dato"})
> db.editoras.find({pais:"Argentina"})


# Hacer una búsqueda donde las editoras sean de Argentina y únicamente mostrar su
# nick
# db.COLECCIÓN.find({campo:"Dato"}, {campoMostrado:1})
> db.editoras.find({pais:"Argentina"}, {nick: 1})
```

También puedes ordenar tus documentos de acuerdo al campo especificado ya sea ascendente (1) o descendente (-1):

```
# Ordenar los resultados de forma ascendente de acuerdo al nombre
# db.COLECCIÓN.find({campo:"Dato"}).sort({campoOrdenado: 1})
> db.editoras.find().sort({nombre:1})

# Ordenar los resultados de forma descendente de acuerdo al nombre
# db.COLECCIÓN.find({campo:"Dato"}).sort({campoOrdenado: -1})
> db.editoras.find().sort({nombre:-1})
```

Modificar documentos

Hasta este momento has visto como crear una base de datos en MongoDB, insertar documentos y hacer búsquedas. Ahora veremos como actualizar campos en tus documentos:

```
# Actualizar un documento
```

```
# db.COLECCIÓN.update({campo:"Dato"},{"$set":{campoModificado:"Dato"}})
> db.editoras.update({nombre:"Yesi"},{"$set":{pais:"Mexico"}})
```

Eliminar documentos y colecciones

Por último es necesario conocer como eliminar un documento de nuestra colección y cómo eliminar también, la colección en sí misma:

```
# Eliminar un documento donde el país sea México
# db.COLECCIÓN.remove({campo:"Dato"})
> db.editoras.remove({pais:"Mexico"})

# Eliminar una colección
# db.COLECCIÓN.drop()
> db.editoras.drop()
```

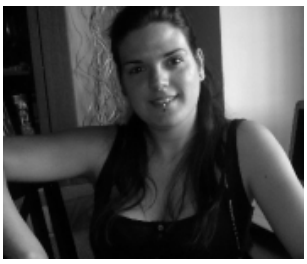
Si deseas saber el total de registros de una colección usa `count()` de la siguiente manera **`db.COLECCION.count()`**

Debido a que es un artículo de primeros pasos con MongoDB explicaré en otro artículo más, operadores para realizar consultas más complejas y cómo podemos ejecutar *scripts* que nos faciliten insertar, modificar, consultar e incluso eliminar datos.

Introducción a Perl (Parte I)

Perl es un gran lenguaje muy utilizado en la administración de sistemas. A partir de esta edición, damos inicio al Manual de Perl. Cada mes una nueva entrega, para que puedas inmiscuirte en el mundo de este maravilloso y legendario lenguaje.

Escrito por: **María José Montes Díaz** (Archera & Programadora)



Estudiante de Grado Ingeniería en Tecnología de la información. Técnico en informática de gestión. Monitora FPO. Docente de programación Python y Scratch para niños de 6-12 años. Activista del software libre y cultura libre.

Webs:
Blog: http://archninfa.blogspot.com.es/

Redes sociales:
Twitter: **@MMontesDiaz**

Perl (*Practical Extraction and Report Languaje*) fue creado por Larry Wall en 1987. Es un lenguaje de programación de distribución gratuita. Su licencia es dual: Artistic License[16] y GPL[17] . Toma características de C, Bourne Shell (sh), AWK, sed y Lisp. Es multi-paradigma: imperativo, funcional, orientado a objetos y reflexivo y además, es un lenguaje interpretado.

Se pensó originalmente para la manipulación de texto. Sin embargo, en la actualidad, se utiliza en múltiples tareas, sobretodo en administración de sistemas y hasta hace poco menos de una década (antes de la existencia de PHP), en el Desarrollo de Aplicaciones Web.

Aunque Perl es un lenguaje interpretado, es muy rápido comparando con otros lenguajes como Burne Shell. Esto se debe a cómo se ejecuta un programa, que podríamos dividir en:

1. **Tiempo de compilación:** Donde se analiza *(parsing)* el texto del programa en un árbol sintáctico. Este árbol es optimizado simplificando las expresiones constantes, propagando el contexto y optimizando partes sueltas del código.

2. **Tiempo de ejecución:** Se ejecuta el programa siguiendo el árbol generado en la compilación.

16 http://www.perlfoundation.org/artistic_license_1_0
17 http://www.gnu.org/licenses/gpl.html

Ejecución de Scripts en Perl

Disponemos de dos métodos:

1.- Añadimos al principio del script la línea:

```
#!/usr/bin/perl
```

Con esta sentencia, indicamos al sistema operativo que debe utilizar Perl (el binario estaría en `/usr/bin/`) para ejecutar lo que viene a continuación. Esta sentencia no es propia de Perl, si no que es propia de sistemas *nix.

2. Especificando el intérprete desde la línea de comando:

```
perl script.pl
```

El primer parámetro que le pasamos a intérprete es el nombre del *script* (archivo `.pl`). Si le pasamos más, serán tomados cómo parámetros del *script*.

El *script*, para poder ser ejecutado sin necesidad de utilizar el segundo método, debe tener permisos de ejecución:

```
chmod +x script.pl
```

Algunas opciones del intérprete de Perl:

```
perl -v
Muestra la versión del intérprete de Perl que estamos utilizando

perl -V
Muestra información sobre la configuración del intérprete de Perl

perl -e expresión
Ejecuta la expresión. Ejemplo:
perl -e "print 'hola mundo\n';"

perl -ne expresión
Ejecuta la expresión a modo de bucle. Por ejemplo: Supongamos que tenemos un
archivo prueba.txt con varios nombres, separados por saltos de línea. Para extraer
aquellos nombres que empiecen por A, podríamos escribir:
perl -ne "print if /A+/;" prueba.txt

perl -pe expresión:
Ejecuta la expresión a modo de bucle, imprimiendo cada línea.
```

Basándonos en el ejemplo anterior, esta sentencia listará el contenido del archivo `prueba.txt`, imprimiendo dos veces las líneas que comiencen por A:

```
perl -pe "print if /A+/;" prueba.txt

perl -d script
Ejecuta el script bajo el depurador.

perl -w script
Da avisos sobre las construcciones con errores.

perl -x script
Empieza a interpretar el archivo que contiene el script cuando encuentra la
referencia al intérprete, por ejemplo: #!/usr/bin/perl.

perl -i archivo
Permite editar archivos. Con la siguiente sentencia, cambiaremos las A por B que
aparezcan en el archivo prueba.txt:
perl -p -i -e "s/A/B/g;" prueba.txt
```

Nuestro primer programa en Perl.

Los *scripts* de Perl son una sucesión de instrucciones, delimitadas por punto y coma (;). Los comentarios se preceden por el símbolo #. A partir de ese símbolo y, hasta el final de línea, se ignora el texto.

Para crear nuestro primer "hola mundo", creamos un archivo, llamado `hola.pl`, con el siguiente contenido:

```
#!/usr/bin/perl
print "Hola, mundo!";
```

Le damos permisos de ejecución:

```
chmod +x hola.pl
```

Y ya podemos ejecutarlo:

```
$./hola.pl
```

NOTA: En la mayor parte de distribuciones GNU/Linux, la ubicación del binario de Perl es `/usr/bin/`. En caso contrario, se deberá especificar la ubicación correcta.

Variables

Una característica de Perl, tomada de la programación Shell, es la marca de las variables con un sigilo precedente. De esta forma, las variables quedan identificadas inequívocamente, permitiendo una sintaxis muy rica y además, permitiendo interpolar variables dentro de las cadenas de caracteres.

Escalares ($variable)

Las variables escalares son las variables básicas de Perl. Pueden contener cualquier tipo de dato: números, letras, cadenas de texto, etc.

La definición de una variable escalar se hace anteponiendo un $ al nombre:

```
$variable
```

El nombre de las variables sólo puede contener caracteres alfanuméricos. Es sensible a mayúsculas y minúsculas. Además, el nombre debe comenzar con letras o _. Después, cualquier carácter alfanumérico puede utilizarse.

En la definición de variables, suele utilizarse la palabra reservada “my”.

Ejemplos:

```
my $numero = 223;
```

O bien, definirla:

```
my $numero;
```

Para después, inicializarla:

```
$numero = 223;
```

Otro ejemplo: definimos la variable $cadena1 y $cadena2 con el texto “cadena”:

```
my $cadena1 = "texto";
my $cadena2 = $cadena1;
```

Arrays (@array)

En Perl, los *arrays* (matrices) son una lista de datos, que pueden ser números, letras, cadenas o variables.

Para definirlo, antepondremos al nombre el símbolo @. La lista de valores irá entre paréntesis y, cada valor, separado por comas (,). El índice de la lista empieza por 0.

```
my @array;
@array (1,2,3,4,5);
```

Para acceder a cada elemento, lo haremos utilizando notación escalar y refiriéndonos a cada elemento por su posición entre []:

```
my $valor=$array[0];
```

Podemos obtener, si es necesario, una sublista, con varios elementos:

```
my @valores=@array[1,3,4];
```

Dónde, el array @valores será inicializado con la lista (2,4,5).

Para obtener el índice del último elemento de la lista, utilizaremos:

1.- Contexto escalar:

```
$numero = @array;
```

2.- Sigilo $#

```
$#array;
```

Para calcular la cantidad de elementos de una lista, bastará sumar 1 al índice del último elemento de la misma:

```
$numero = @array+1;
```

Hashes (%hash)

Este tipo de variable, permite asociar cada elemento de una lista a una llave. Pocos lenguajes incorporan un tipo similar a éste.

El sigilo utilizado en este caso es el %. Se pueden definir de dos maneras:

```
my %hash = (u,'uno',d,'dos',t,'tres');
```

O bien:

```
my %hash = (
    u ==> 'uno',
    d ==>'dos',
    t ==>'tres',
);
```

Para llamar al valor de una de nuestras llaves la sintaxis es:

```
$hash{nombre_llave};
```

Por ejemplo:

```
$hash{u};
```

Igual que hicimos con los *arrays*, podemos obtener una lista, cambiando el contexto, mediante sigilos, de la siguiente forma:

```
@lista=@hash{u,t}
```

Dónde, @lista, tomará los valores ('uno','tres').

FILEHANDLES

Es un tipo de variables utilizadas para el acceso a archivos.

DIRHANDLES

Igual que el anterior tipo, pero para carpetas.

Introducción al desarrollo dirigido por pruebas

Mucho se ha hablado de una de las doce prácticas técnicas más beneficiosas de la Programación eXtrema, pero a excepción del excelente libro *«Diseño Ágil con TDD*[18]*»* de *Carlos Blé Jurado*, muy poca es la bibliografía que podremos encontrar en español. Y si a esto le sumamos lo complejo que resulta asimilar el tema, se hace evidente la necesidad de sumar textos en nuestra lengua, que nos ayuden a comprender la importancia de esta técnica y aprender a implementarla.

Escrito por: **Eugenia Bahit** (Arquitecta GLAMP & Agile Coach)

Eugenia es **Arquitecta de Software**, **docente** instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y eXtreme Programming. Miembro de la **Free Software Foundation** e integrante del equipo de **Debian Hackers**.

Webs:
Cursos de programación a Distancia: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

TDD (siglas de *Test-Driven Development* - Desarrollo conducido por pruebas) es una **técnica de programación** que consiste en guiar el desarrollo de una aplicación, por medio de **Test Unitarios**. Los Test Unitarios (Unit Test, en inglés) no son más que algoritmos que emulan lo que la aplicación se supone debería hacer, convirtiéndose así, en un modo simple de probar que lo que "piensas" programar, realmente funciona.

A la vez, esta técnica te permitirán saber: qué, cómo, cuáles y cuántos algoritmos necesitarás desarrollar para que tu aplicación haga lo que realmente debe hacer y no falle, ante ciertos casos que no son los que se esperaban.

18 http://www.dirigidoportests.com/el-libro

> *El objetivo del TDD no es "adivinar", sino probar y actuar, ya que los Test Unitarios serán una guía para entender como funciona el código, ayudándote a organizarlo de manera clara, legible y simple, a la vez de servir como "documentación"*

Carlos Blé Jurado en su libro **Diseño Ágil con TDD** nos define la técnica de TDD como:

> *"[...] la respuesta a las grandes preguntas: ¿Cómo lo hago? ¿Por dónde empiezo? ¿Cómo se qué es lo que hay que implementar y lo que no? ¿Cómo escribir un código que se pueda modificar sin romper funcionalidad existente? [...]"*

Según **Kent Beck** -uno de los co-fundadores de la Programación eXtrema (XP)-, implementar TDD nos otorga seis grandes ventajas:

1. **La calidad del software aumenta** disminuyendo prácticamente a cero, la cantidad de *bugs* en la aplicación;
2. Conseguimos **código altamente reutilizable** puesto que los test nos obligan a desarrollar algoritmos genéricos;
3. **El trabajo en equipo se hace más fácil**, une a las personas, ya que al desarrollar con test, nos aseguramos de no romper funcionalidades existentes de la aplicación;
4. Nos permite **confiar en nuestros compañeros de equipo** aunque tengan menos experiencia. Esto es, debido a que el hecho de tener que desarrollar test antes de programar el algoritmo definitivo, nos asegura -independientemente del grado de conocimiento y experiencia del desarrollador- que el algoritmo, efectivamente hará lo que se supone debe hacer y sin fallos;
5. Escribir el ejemplo (test) antes que el código **nos obliga a escribir el mínimo de funcionalidad necesaria, evitando sobre-diseñar**, puesto que desarrollando lo mínimamente indispensable, se obtiene un panorama más certero de lo que la aplicación hace y cuál y cómo es su comportamiento interno;
6. **Los tests son la mejor documentación técnica** que podemos consultar a la hora de entender qué misión cumple cada pieza del rompecabezas, ya que cada test, no es más que un "caso de uso" traducido en idioma informático.

Características de los Test Unitarios

Los Test Unitarios (o técnica de *Unit Testing*), representan el alma de la programación dirigida por pruebas. Son algoritmos independientes que se encargan de verificar -de manera simple y rápida- el comportamiento de **una parte mínima de código**, de forma **individual**, **independiente** y **sin alterar el funcionamiento de otras partes de la aplicación**.

Un Test Unitario posee **cuatro características particulares** que debe conservar a fin de considerarse "unitario". Estas son:

1. **Atómico:**

   Prueba una parte mínima de código.

   Dicho de manera simple, cada test unitario debe probar una -y solo una- "acción" realizada por un método.

   Por ejemplo, para un método que retorna el neto de un monto bruto más el IVA correspondiente, deberá haber un test que verifique recibir en forma correcta el importe bruto, otro que verifique el cálculo del IVA sobre un importe bruto y finalmente, un tercer test unitario que verifique el cálculo de un importe bruto más su IVA correspondiente.

2. **Independiente:**

   Cada Test Unitario DEBE ser independiente de otro.

   Por ejemplo, siguiendo el caso anterior, el test que verifique la suma de un importe bruto más su IVA correspondiente, no debe depender del test que verifica el cálculo del IVA.

3. **Inocuo:**

   Podría decirse que cada test unitario debe ser inofensivo para el sistema.

   Un test unitario DEBE poder correrse sin alterar ningún elemento del sistema, es decir, que no debe, por ejemplo, agregar, editar o eliminar registros de una base de datos.

4. **Rápido:**

   La velocidad de ejecución de un test unitario cumple un papel fundamental e ineludible en el desarrollo guiado por pruebas, ya que de la velocidad de ejecución de un test, dependerá de manera proporcional, la velocidad con la que una funcionalidad se desarrolle.

Anatomía de un Test

Los Test Unitarios se realizan, en cualquier lenguaje de programación, mediante herramientas que proveen un completo entorno de trabajo (*Frameworks* para *Unit Testing*). Estos entornos de trabajo, son desarrollados con un **formato** determinado, conocido como **xUnit**.

De allí, que los *frameworks* para *Unit Testing* que cumplen con dicho formato, suelen tener nombres compuestos por una abreviatura del lenguaje de programación, seguida del término "*unit*". Por ejemplo: **PyUnit** (Python), **PHPUnit** (PHP), **ShUnit** (Shell Scripting), etc.

Exceptuando el caso de Shell Scripting, **los frameworks xUnit de lenguajes que soportan la orientación a objetos, utilizan éste paradigma** tanto en su anatomía de desarrollo como para su implementación (creación de los test unitarios).

Por lo tanto, los Test Unitarios se agrupan en **clases**, denominadas **Test Case**, que heredan de una clase del *framework* xUnit, llamada **xTestCase**:

```
# Ejemplo con PyUnit
import unittest

class BalanceContableTestCase(unittest.TestCase):
    pass


# Ejemplo con PHPUnit
class BalanceContableTest extends PHPUnit_Framework_TestCase {
}
```

Los métodos contenidos en una clase Test Case, pueden o no, ser Test Unitarios. **Los Test Unitarios** contenidos en una clase Test Case, **deben contener el prefijo `test_` en el nombre del método** a fin de que el *framework* los identifique como tales:

```
# Ejemplo con PyUnit
import unittest

class BalanceContableTestCase(unittest.TestCase):

    def test_calcular_iva(self):
        pass


# Ejemplo con PHPUnit
class BalanceContableTest extends PHPUnit_Framework_TestCase {

    public function test_calcular_iva() {
    }

}
```

Otra ventaja que los *frameworks* xUnit nos proveen, es la facilidad de poder crear **dos métodos especiales** dentro de una clase Test Case **-que no son test-**, los cuales están

destinados a **preparar el escenario** necesario para correr los test de esa clase y **eliminar aquello que se desee liberar**, una vez que el test finalice. Estos métodos, son los denominados **`setUp()`** y **`tearDown()`** respectivamente:

```
# Ejemplo con PyUnit
class BalanceContableTestCase(unittest.TestCase):

    def setUp(self):
        self.importe_bruto = 100
        self.alicuota_iva = 21

    def tearDown(self):
        self.importe_bruto = 0
        self.alicuota_iva = 0

    def test_calcular_iva():
        pass


# Ejemplo con PHPUnit
class BalanceContableTest extends PHPUnit_Framework_TestCase {

    public function setUp() {
        $this->importe_bruto = 100;
        $this->alicuota_iva = 21;
    }

    public function tearDown() {
        $this->importe_bruto = 0;
        $this->alicuota_iva = 0;
    }

    public function test_calcular_iva() {
    }

}
```

Los métodos `setUp()` y `tearDown()` se ejecutan antes y después de cada test, respectivamente.

Cada Test estará dividido a la vez, en **tres partes** identificadas por las siglas **AAA** las cuáles representan a las tres "acciones" que son necesarias llevar a cabo, para dar forma a los Tests: ***Arrange***, ***Act*** *and* ***Assert*** (preparar, actuar y afirmar).

Preparar consiste en definir y configurar (dentro del test) los recursos necesarios para poder **actuar**, es decir, hacer la llamada al código del **Sistema cubierto por Test** (**SUT**) que se desea probar. Esto se conoce como "cobertura de código" o *Code Coverage* (o *Coverage* a secas).

Finalmente, **afirmar** el resultado de un test, se refiere a invocar al método `assert` del *Framework* xUnit, que sea necesario para "afirmar que el resultado obtenido durante la actuación, es el esperado".

Los métodos `assert` (métodos de afirmación), son métodos que vienen definidos por defecto en el *framework* xUnit, destinados a verificar un resultado. Los nombres de estos métodos suelen ser muy similares entre los diversos *frameworks*, como por

ejemplo `assertTrue()` para verificar que el valor del resultado de la actuación, devuelto por el SUT, sea `True`. Pero los veremos en detalle en la próxima entrega.

Si quieres ir conociendo los métodos assert, puedes ir visitando las referencias oficiales: **PyUnit:** http://docs.python.org/2/library/unittest.html#test-cases y **PHPUnit:** http://www.phpunit.de/manual/current/en/writing-tests-for-phpunit.html#writing-tests-for-phpunit.assertions

Por favor, notar que mientras que PyUnit viene incluido como un módulo estándar de Python, PHPUnit, requiere de instalación por separado. En la próxima entrega, abarcaremos también la instalación de PHPUnit.

Algoritmo para las pruebas unitarias

Existe un algoritmo para escribir Test Unitarios, el cual consiste en:

- **PRIMER PASO:** Escribir el Test y hacer que falle (retornando en el SUT, un valor contrario al que se espera que devuelva en realidad)
- **SEGUNDO PASO:** Escribir la mínima cantidad de código necesaria (en el SUT) para que el test pase.
- **TERCER PASO:** Escribir un nuevo test (para el mismo SUT) y esperando que falle el nuevo test.
- **CUARTO PASO:** Escribir nuevamente, en el SUT, la mínima cantidad de código necesaria para que ambos test pasen (generalmente, la primera vez, para que el test pase, se "*hardcodea*" -se escribe "a mano"- el valor de retorno del SUT y, en esta segunda oportunidad, se escribe el mínimo algoritmo necesario para que el SUT retorne el mismo valor pero sin estar "hardcodeado").

*En la **siguiente entrega**, nos enfocaremos en los Test Unitarios con **PyUnit** y **PHPUnit**.*

IPv6, el presente

El crecimiento exponencial de Internet, ha hecho que desde los años '80, el agotamiento de las direcciones IPv4 haya sido un tema de preocupación. El agotamiento ya ha llegado y nos encontramos en la era del IPv6.

Escrito por: **Laura Mora** (Administradora de Redes y Sistemas GNU/Linux)

Laura es administradora de **Redes y Sistemas GNU/Linux** en Cataluña. Tiene una larga trayectoria en la consultoría de soluciones telemáticas **opensource** para movimientos sociales. La base de sus proyectos es el **empoderamiento tecnológico** de las personas.

Webs:
Blog: http://blackhold.nusepas.com/
Web: http://delanit.net

Redes sociales:
Twitter / Identi.ca: **@Blackhold_**

El 31 de Enero de 2011, la **IANA (Internet Assigned Numbers Authority)**, entidad que se encarga de distribuir las IP en Internet, asignó los dos últimos bloques /8 (16 millones de IP) y el 3 de Febrero finalmente se asignaron los últimos 5 bloques de direcciones disponibles a los cinco registros de Internet (**Regional Internet Registries o RIR**), que tienen delegada la asignación en las cinco regiones administrativas en que se divide el mundo según Internet. Aquí podemos ver[19] cuando los RIR van a agotar definitivamente las Ipv4 disponibles.

En 1999 se preveía que si la red seguía creciendo al ritmo que lo hacía, el numero de IP se agotaría antes de encontrar una solución al problema, así que se diseñó **NAT (Network Address Translation)**. El concepto de NAT consiste en utilizar una dirección IP enrutable (o un número limitado de direcciones IP) para conectar todas las máquinas a través de la traducción, en la pasarela de Internet, entre la dirección interna (no enrutable) de la máquina que se desea conectar y la dirección IP de la pasarela.

Además, el proceso de traducción de direcciones permite a las compañías asegurar la red interna siempre y cuando oculte la asignación de direcciones internas. Para un observador que se ubica fuera de la red, todos los pedidos parecen provenir de la misma dirección IP.

NAT ha podido parecer un gran invento, también una forma de "tener los ordenadores a salvo del salvaje Internet". NAT ha sido adoptado como firewall, pero en realidad no lo es ya que se permite que protocolos como UPNP o conexiones inversas puedan acceder

19 http://www.potaroo.net/tools/ipv4/index.html

a equipos detrás del NAT.

Sin embargo, este sistema no está facilitando la tarea a todas aquellas aplicaciones de tipo cliente-servidor (que ofrecen servicios dedicados a los usuarios), o permitiendo las cliente-cliente, tal como podría ser VoIP, Televisión Interactiva, juegos on-line, programas P2P, etc. Si somos usuarios de estos servicios seguramente alguna vez habremos tenido que acceder a nuestro router y configurar la redirección de puertos. Con IPv6 ¡esta farragosa tarea desaparece! cada ordenador dispone de una IP global y con esto, es posible rutearlo directamente a Internet, así que desde hace ya unos años, los sistemas operativos se toman mucho más en serio el tema de la seguridad y por un buen motivo, un sistema o aplicación que no esté preparado para las redes del futuro, ¡estará condenado a desaparecer!

Además de esto nos encontramos con la realidad del imparable crecimiento de Internet, que en los últimos 10 años ha experimentado un crecimiento superior al 600%, puesto que día tras día se unen a esta gran red, nuevos tipos de dispositivos.

IPv6, el protocolo

Este protocolo empezó a ser diseñado en 1996 por la **IETF (Internet Engineering Task Force)**[20]**,** organización que desarrolla y promueve estándares -sobretodo- acerca de TCP/IP y protocolos de Internet. Desde entonces IPv6 ha ido evolucionando, pero a diferencia de otros protocolos se decidió no definir versiones, en motivo del farragoso proceso de estandarización de los RFC que hace que muchos estándares se apliquen en la red antes de ser estandarizados, como por ejemplo el polémico 802.11n.

Uno de los cambios que saltan mas a la vista es la notación de las direcciones, que trata de direcciones de 128bits de longitud escritos en 8 grupos de cuatro dígitos hexadecimales:

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7334
```

Esto es una dirección IPv6 válida. Usamos el prefijo 2001:0db8: para documentación.

Si un grupo de cuatro dígitos es nulo (que tiene el valor 0000), puede ser comprimido, así que estas dos direcciones IPv6 serían en realidad la misma:

```
2001:0db8:85a3:0000:1319:8a2e:0370:7344
2001:0db8:85a3::1319:8a2e:0370:7344
```

Siguiendo esta regla, si más de dos grupos consecutivos son nulos pueden comprimirse como "::". Si la dirección tiene más de una serie de grupos nulos consecutivos, la compresión se puede aplicar en más de uno, así que:

20 http://www.ietf.org/

```
2001:0DB8:0000:0000:0000:0000:1428:57ab
2001:0DB8:0000:0000:0000::1428:57ab
2001:0DB8:0:0:0:0:1428:57ab
2001:0DB8:0::0:1428:57ab
2001:0DB8::1428:57ab
```

Son todas válidas y representan lo mismo. Pero:

```
2001::25de::cade
```

no es válida porqué no queda claro cuantos grupos nulos hay en cada lado.

Los ceros iniciales en un grupo pueden ser omitidos. Así que:

```
2001:0DB8:02de::0e13
```

es lo mismo que:

```
2001:DB8:2de::e13
```

Tipos de direcciones IPv6

- **Multicast** (una a muchas): "ff00::/8"
- **Anycast** (una a la más cercana): para la réplica de DNS (por ejemplo para hacer peticiones al servidor DNS más cercano topológicamente hablando).
- **Unicast**:
  - Globales : equivaldrían a lo que conocemos como IP públicas en IPv4.
  - Enlace local (link-local): "fe80::/10". Sólo son válidas a nivel 2. serían como las 169.254.0.0/16 en IPv4. No son enrutables en la red IPv6. Más información[21].
  - Locales únicas (ULA): "fc00::/7". Estas tampoco son enrutables en la red IPv6 y su uso común es para redes privadas virtuales. Un ejemplo podría ser el caso de un banco, que tiene sus ordenadores conectados a Internet con su IP global, pero el programa de gestión del banco sólo sería accesible desde la ULA. Es habitual pues, que un ordenador con IPv6 disponga de varias direcciones IPv6. Más información[22].
  - IPv4 mapeadas: "::ffff:0:0/96" (en desuso)

21 http://es.wikipedia.org/wiki/Dirección_de_Enlace-Local
22 http://en.wikipedia.org/wiki/Unique_local_address

- Compatible IPv4: (desaprobadas)
- Local de sitio (site-local): (desaprobadas) "fec0::/10": para definir una red interna en un edificio o planta.
- Loopback: "::1/128" el equivalente a la 127.0.0.1/8

Los rangos de IP

IPv6 está diseñado para que cada máquina tenga asignado un rango en lugar de una única IP. Solo en casos muy concretos se aconseja que un dispositivo electrónico tenga una única IP.

El rango mínimo que se debería asignar a una interfaz de red es un /64 (2^64 IP).

De esta forma puede usarse la MAC address del dispositivo físico para generar una IP válida en esos restantes 64 bits. Así, entre otras cosas, se gana privacidad, pues ya no será trivial descubrir la IP de una máquina y escanear el rango entero nos podría costar varios años.

Los RIR recomiendan a los ISP que entreguen a sus clientes un /48. De esta forma el cliente puede asignar hasta 2^16 /64 a sus dispositivos domésticos (nevera, teléfono, televisor, enchufe, persiana...)

> *Asignando un /48 a cada persona del planeta, suponiendo que estas /48 no se devolviesen, se podría asignar un /48 a cada persona del planeta durante 480 años.*
> *¡Se espera que el protocolo quede obsoleto mucho antes!*

Broadcast y ARP

Otro cambio realmente significativo, es que IPv6 no implemente *Broadcast* y, por lo tanto, no existe *broadcast domain*.

En su lugar se ha extendido y mejorado el funcionamiento de *Multicast*. Para comunicar los ordenadores directamente conectados (y sin conocer previamente sus IP) se puede usar la dirección link-local multicast `ff02::1`.

En IPv6 ARP, ha sido sustituido por un protocolo mucho más potente: NDP (Neighbor Discovery Protocol[23]), que usa ICPMv6 para sus comunicaciones.

23 http://en.wikipedia.org/wiki/Neighbor_Discovery_Protocol

Una de las consecuencias de estos dos cambios es que, por ejemplo, se pueden comunicar dos máquinas conectadas sin necesidad de configurar ninguna IP. Las direcciones link-local (`fe80::/10`), que se configuran automáticamente en nuestro sistema, nos lo permiten:

```
ssh usuario@ipv6link-local%eth0
```

Como se puede apreciar, la interfaz de red debe ser especificada ya que sino, el kernel no tiene forma de saber hacia donde enviar los paquetes. El ordenador destino verá que la dirección de destino es su propia IP link-local y aceptará el paquete.

ULA

También funcionaría por las ULA, pero no tiene sentido, ya que aunque no sean enrutables por Internet, sí pueden serlo dentro de la red local y en este caso, el *gateway* ya nos enrutaría a la ubicación correcta.

```
Para añadir una ULA:
ip -6 addr add ULA/64 dev INTERFACE
```

El enrutamiento estático

El comportamiento de la red en esta versión cambia de forma significativa con la versión anterior. Por ejemplo, no es usual encontrar configuraciones como la siguiente:

```
eth0: 192.168.1.1/24
eth1: 192.168.1.10/16
```

En IPv6, lo habitual es que un mismo rango esté incluido en dos dispositivos de red (uno incluye el otro), el más pequeño es siempre el que manda. Esta característica fue exportada a IPv4.

```
2001:0DB8:1006::/64
es mas pequeño que:
2001:0DB8:1006:1100::/56
```

SLACC (Stateless Address Autoconfiguration)

En IPv6 son los mismos *hosts* quienes se configuran automáticamente usando el protocolo Neighbor Discovery Protocol[24] (NDP) mediante mensajes ICMPv6[25].

24 http://en.wikipedia.org/wiki/Neighbor_Discovery_Protocol
25 http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol_version_6

En el momento que éstos se conectan a la red, envían al router una solicitud de link-local mediante una petición *multicast (router solicitation)* pidiendo los parámetros de configuración. Entonces, los routers responderán con un paquete *"router advertisment"* que contiene los parámetros de configuración de la red.

La cabecera de IPv6

IPv6 incrementa la longitud de la cabecera IP de 20 a 40 bytes. La cabecera IPv6 contiene dos direcciones de 16 bytes (fuente y destino) precedidas de 8 bytes de control. La reducción de la información de control y la eliminación de opciones de la cabecera tienen como fin optimizar el procesamiento del paquete. Los campos de uso poco frecuente se han eliminado de la cabecera y se han pasado a extensiones de cabecera opcionales.

Offset del Octeto		0								1								2								3							
	Bit Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Versión				Clase de Tráfico								Etiqueta de Flujo																			
4	32	Longitud del campo de datos																Cabecera Siguiente								Límite de Saltos							
8	64	Dirección de Origen																															
C	96																																
10	128																																
14	160																																
18	192	Dirección de Destino																															
1C	224																																
20	256																																
24	288																																

El campo "cabecera siguiente" o *"next header"*, permite añadir nuevos campos en la cabecera sin tener que actualizar el software o el hardware de todos los dispositivos de la red. Y es que IPv6 se ha diseñado para poder implementar nuevos cambios de forma progresiva sin tener que apagar toda la red para aplicarlos. Más información[26].

La asignación de IP

En IPv6 exististe DHCPv6, pero no es la opción recomendada. Para ello tenemos el nuevo protocolo SLAAC.

En el router cabecera activamos el RA y, los equipos de la red al conectarse a ella, ven el anuncio del router y son ellos mismos quienes se aplican la dirección IP a partir de, por ejemplo, su dirección MAC. En sistemas basados en GNU/Linux siempre se asigna dicha IP a partir de la MAC, pero en otros sistemas existen las llamadas direcciones de

26 http://en.wikipedia.org/wiki/IPv6_Packet

privacidad[27], donde se genera la IP de forma aleatoria.

Para activar IPv6 privacy en GNU/Linux deberemos activar las *"IPv6 Privacy Extensions"*. Como *root* editamos el archivo `/etc/sysctl.conf` y añadimos estas líneas:

```
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
```

Reiniciamos la red (o la maquina si se resiste al cambio) y entonces, si disponemos de un RA en nuestra red, veremos que nos asigna 2 direcciones IPv6: una basada en nuestra MAC y otra aleatoria. Un ejemplo:

```
wlan0     Link encap:Ethernet  HWaddr 00:21:6a:a5:18:79
          inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: 2001:db8:1006:0:221:6aff:fea5:1879/64 Scope:Global
          inet6 addr: fe80::221:6aff:fea5:1879/64 Scope:Link
          inet6 addr: 2001:db8:1006:0:2c04:41de:e3bc:ffe/64 Scope:Global
```

De esta forma, las peticiones en una página web aparecen como si se hicieran desde la *IP privacy*:

```
Nombre del Ordenador:      2001:db8:1006:0:2c04:41de:e3bc:ffe
```

Una dirección IP, en el caso que tengamos una ip /64, vendrá formada de la siguiente manera:

- los 64 primeros bits vendrán dados por nuestro proveedor
- los últimos 64, se generarán "aleatoriamente".

Al igual que en IPv4, podremos usar igualmente DHCP (aunque no sería recomendable) o bien definir una IPv6 de forma manual.

La recomendación con IPv6 (y esto es realmente muy importante), seria olvidarse de recordar un montón de direcciones IP y empezar a usar los **DNS (Domain Name System)**, que ¡para esto fueron creados! :)

En los DNS, para asignar una dirección IPv4 usamos el Address Record[28] A, en IPv6 usaremos AAAA, así que al ejecutar una petición de resolución sobre un dominio nos va a devolver las direcciones IPv4 y IPv6 relacionadas. Si nuestra red tiene salida IPv6, la dirección IPv6 será la prioritaria:

27 http://www.ietf.org/rfc/rfc3041.txt
28 http://en.wikipedia.org/wiki/List_of_DNS_record_types

```
$ host guifi.net
guifi.net has address 109.69.8.5
guifi.net has IPv6 address 2a00:1508::5
```

El protocolo de movilidad

En IPv6 se ha desarrollado un conjunto de protocolos y herramientas de movilidad[29], que permiten "saltar" de red a red sin perder en ningún momento la conectividad. Por ejemplo, gracias a SLAAC, los últimos 64 bits de nuestra IP son fijos (basados en la MAC). Así que si sabemos el prefijo (los primeros 64 bits) de la nueva red donde nos vamos a mover, podemos hacerlo sin perder nunca la conexión.

Por ejemplo, estamos en nuestra casa haciendo una llamada telefónica con nuestro móvil conectado a nuestra wi-fi y de repente salimos a la calle, fuera del alcance de nuestra wi-fi y pasamos a la red móvil de nuestro operador telefónico. Gracias a este protocolo podremos realizar dicha llamada telefónica ¡sin ningún corte!

Extensión IPSec

IPv4 se diseñó para ser funcional, no seguro. Pero con IPv6 se ha dado un paso adelante en este aspecto. Es la razón por la que se ha implementado IPSec en este protocolo. El uso de IPSec no es obligatorio en IPv6, lo que sí es realmente obligatorio es que todos los dispositivos que soporten IPv6 (y esto sería extensivo para las conexiones end2end[30]), también soporten IPSec, dejando a nuestro libre albedrío la decisión de proteger o no nuestras comunicaciones. Más información[31]

Trabajando con IPv6

La primera impresión que tenemos con IPv6 es pensar... ¡qué jaleo! ¡se han vuelto locos! ¡esto es el fin! Pero en realidad migrar una red de IPv4 a IPv6 no es una cosa tan crítica, además de que ambos protocolos ¡pueden convivir perfectamente sin ningún tipo de conflicto! Y es que TCP/IP aunque sea el protocolo más extendido no es el único. Posiblemente si analizamos nuestra red y se ha conectado nuestro vecino con su «*icacharro*», podremos ver peticiones del protocolo *«manzanatalk»*. Vale que *«manzanatalk»* cada vez está más en desuso, pero es el primer protocolo distinto a tcp/ip que nos viene a la cabeza. Existen otros como netbios, ipx, etc.

Posiblemente nuestro operador de telecomunicaciones no está haciendo su trabajo y en tu casa aún no dispones de IPv6, pero ¡no desesperes! ¡hay opción!

Existen los llamados tunnel brokers[32], que te permiten realizar túneles 6to4 (sit). Todos los que he probado requerían IPv4 estática, pero parece que existe ya, alguno que

29 http://long.ccaba.upc.es/long/050Dissemination_Activities/tomas_demiguel.pdf
30 http://en.wikipedia.org/wiki/End-to-end_principle
31 http://www.ipv6.com/articles/security/IPsec.htm
32 http://en.wikipedia.org/wiki/List_of_IPv6_tunnel_brokers

también permiten IPv4 dinámica.

Así que vayamos a ver uno de los mas conocidos: el de *hurricane electric*, que además te ofrece los comandos para hacer un *copy-paste* en tu consola y disponer al acto de IPv6 :)

Hurricane Electric (tunnelbroker.net)

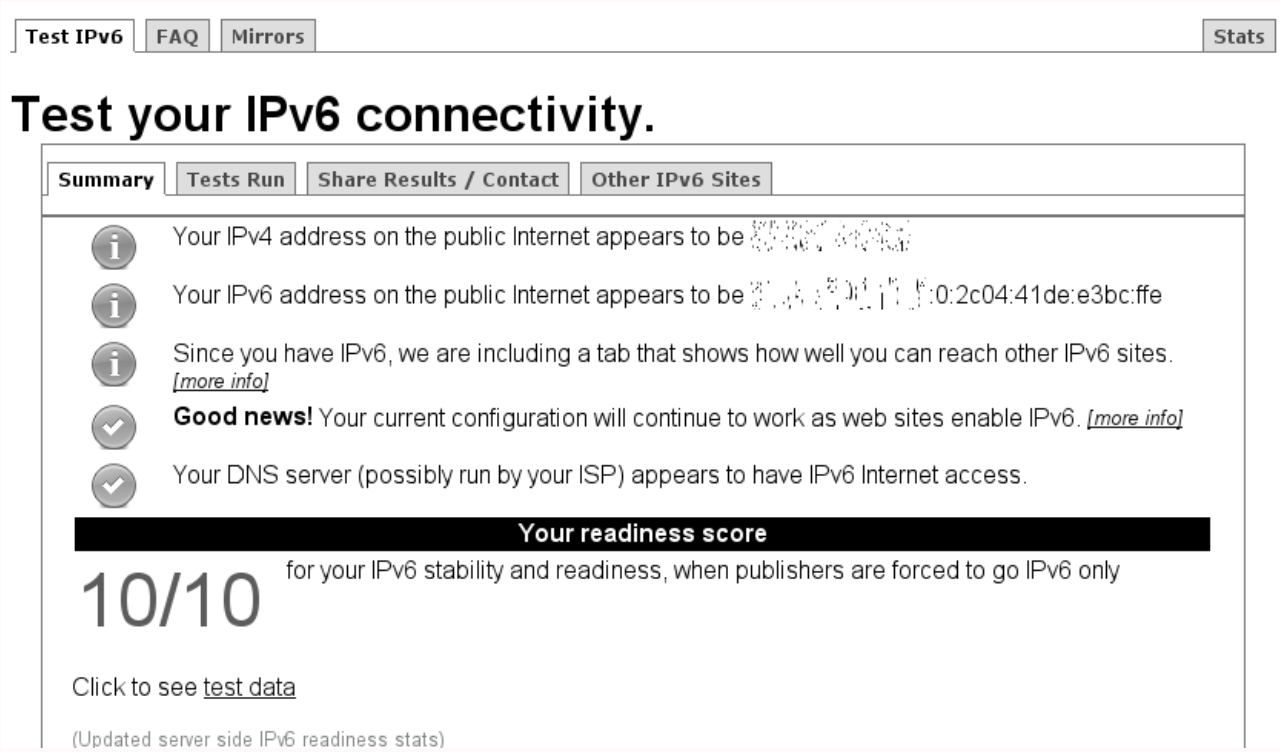
Esta página requiere registro y también IPv4 estática, pero una vez pasamos estos trámites podemos crear un túnel 6to4 realmente estable.

Una vez *logueados* vamos a *"Create Regular Tunnel"*, especificamos nuestra dirección IP (si no la ha localizado), escogemos uno de los servidores y le damos a *"Create Tunnel"*.

A continuación, nos aparece una página con nuestra configuración IPv6 y en la pestaña "*example configurations*" escogemos nuestro sistema operativo (el caso más común Linux-net-tools):

```
ifconfig sit0 up
ifconfig sit0 inet6 tunnel ::216.66.84.46
ifconfig sit1 up
ifconfig sit1 inet6 add 2001:470:1f14:18b::2/64
route -A inet6 add ::/0 dev sit1
```

A partir de aquí ya dispondremos de un túnel IPv6 en nuestra máquina y uno de los siguientes pasos es probar si estamos ya preparados para el mundo IPv6: http://test-ipv6.com/



Una vez tengamos IPv6, podemos realizar algunas pruebas. Dichas pruebas están hechas desde un servidor de `giss.tv` (`giss.delanit.net`), así que a partir de aquí las direcciones

IPv6 son reales:

```
$ host giss.delanit.net
giss.delanit.net is an alias for v-giss.delanit.net.
v-giss.delanit.net has address 109.69.9.56
v-giss.delanit.net has IPv6 address 2a00:1508:1000::6d45:938
```

Configuración IP

Este servidor tiene en `eth0` la IPv4 (pública) 108.69.9.56 y la IPv6 (global) `2a00:1508:1000::6d45:938`. En `eth1` la IPv4 interna de guifi.net. En la interfaz `lo` nos fijamos que está la ::1 que equivale a la 127.0.0.1.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:18:51:e6:d6:47
          inet addr:109.69.9.56  Bcast:109.69.9.63  Mask:255.255.255.240
          inet6 addr: 2a00:1508:1000::6d45:938/64 Scope:Global
          inet6 addr: fe80::218:51ff:fee6:d647/64 Scope:Link

eth1      Link encap:Ethernet  HWaddr 00:18:51:16:16:17
          inet addr:10.138.15.131  Bcast:10.138.15.159 Mask:255.255.255.224
          inet6 addr: fe80::218:51ff:fe16:1617/64 Scope:Link

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host

# route -6 -n
Kernel IPv6 routing table
Destination                Next Hop                   Flag Met Ref Use If
2a00:1508:1000::/64        ::                         U    256 0     1 eth0
fe80::/64                  ::                         U    256 0     0 eth0
fe80::/64                  ::                         U    256 0     0 eth1
::/0                       2a00:1508:1000::6d45:901   UG   1024 1   312 eth0
::/0                       ::                         !n   -1  1   830 lo
::1/128                    ::                         Un   0   1 31179 lo
2a00:1508:1000::6d45:938/128   ::                     Un   0   1 21440 lo
fe80::218:51ff:fe16:1617/128   ::                     Un   0   1     0 lo
fe80::218:51ff:fee6:d647/128   ::                     Un   0   1    74 lo
ff00::/8                   ::                         256 0     0 eth0
ff00::/8                   ::                         U    256 0     0 eth1
::/0                       ::                         !n   -1  1   830 lo
```

Al ser un servidor, le hemos definido al igual que la IPv4, una IPv6 estática:

```
[/etc/network/interfaces]
auto lo eth0
iface lo inet loopback
iface eth0 inet static
        address 109.69.9.56
        netmask 255.255.255.240
        gateway 109.69.9.49

iface eth0 inet6 static
        address 2a00:1508:1000::6d45:938
        netmask 64
```

```
        up ip r add default via 2a00:1508:1000::6d45:901 dev eth0
```

Ping

```
$ ping6 guifi.net
PING guifi.net(2a00-1508--5.ip6.guifi.net) 56 data bytes
64 bytes from v6.guifi.net: icmp_seq=1 ttl=62 time=2.47 ms
64 bytes from 2a00-1508--5.ip6.guifi.net: icmp_seq=2 ttl=62 time=1.96 ms
64 bytes from v6.guifi.net: icmp_seq=3 ttl=62 time=2.08 ms
```

Traceroute

```
$ traceroute -6 guifi.net
traceroute to guifi.net (2a00:1508::5), 30 hops max, 80 byte packets
 1  2a00:1508:1000::6d45:901 (2a00:1508:1000::6d45:901)  0.183 ms  0.166 ms  0.280 ms
 2  2a00:1508:ffff:fffb::21 (2a00:1508:ffff:fffb::21)  2.032 ms  2.034 ms  2.029 ms
 3  v6.guifi.net (2a00:1508::5)  2.438 ms  2.430 ms  2.423 ms
```

Si nos fijamos, estamos usando las típicas herramientas de red, pero le estamos añadiendo la opción `-6`.

De momento, como no todo el mundo dispone de IPv6 en sus servidores, dependemos de IPv4 hasta que poco a poco toda la red vaya migrando a IPv6 (algunos proveedores estiman que el proceso no se completará hasta 2030). Muchos, antes de empezar a mirar Ipv6, nos imaginamos que este cambio puede ser una cosa tan crítica como el más que conocido efecto 2000[33], en la que nos pensábamos que todo dejaría de funcionar. Pero como todo, es bueno estar preparados antes de que nos pille el tren y lo más importante, es informarse de lo que realmente implica este cambio. En mi humilde opinión ¡a mejor! :)

Presentes y futuros usos de un Internet basado en IPv6

Al igual que cuando se diseñó IPv4, nadie imaginaba que terminaríamos todos con pequeños dispositivos en nuestros bolsillos con una capacidad computacional de varios cientos de veces de la de aquellos antiguos ordenadores de los años '80.

Pero es que en la actualidad somos incapaces de imaginarnos cómo accederemos a Internet en el futuro inmediato, con qué dispositivos, cuales serán sus usos... etc. Y es que esta realidad la podemos ver sin tener que ir muchos años atrás. ¿Quién hubiese imaginado hace tan sólo 5 años que lo común sería estar con nuestro teléfono móvil sacando fotos de una cosa que nos gusta en la calle y compartiéndola en tiempo real con cientos de amigos/contactos en la red? Las protestas organizadas simultáneamente

33 http://es.wikipedia.org/wiki/Problema_del_año_2000

a varios puntos del planeta han sido posibles gracias a la evolución de la tecnología.

Cuando tenemos a nuestro alcance la posibilidad de asignar una dirección IP a cualquier dispositivo, nos pueden venir a la cabeza algunos de sus usos en modo de ciencia ficción.

Algunos de los usos más novedosos que se le está dando a IPv6 en la actualidad es en el ámbito de la domótica: casas "inteligentes" en las cuales podemos controlar con nuestro dispositivo móvil hasta la mas insignificante bombilla. La imaginación se desborda cuando tratamos temas como la seguridad o la reducción de costes energéticos a favor de nuestro bolsillo y del medio ambiente.

Otro de los posibles usos que podría tener es, por ejemplo, para gestionar mucho mejor nuestra agenda. Cada componente electrónico de nuestro coche podría tener su única dirección IP, estar conectado con la base de datos del fabricante y en el momento de que este fallase, el propio coche contactaría con nuestro taller mecánico, pediría la pieza y el coche se coordinaría con nuestra agenda para que, por ejemplo, cuando tengamos que ir a hacer una visita a un cliente, en lugar de aparcar el coche en el parquímetro dejarlo en el taller mecánico más cercano, dónde nos estaría esperando el nuevo recambio.

No tenemos que olvidarnos tampoco de las ventajas que puede aportar IPv6 en el campo de la medicina, para el seguimiento en tiempo real de enfermos los cuales es preciso tener un control exhaustivo de su tratamiento, así como monitorización remota de dispositivos de electro-medicina, como los marcapasos, control de la diabetes, etc.

Coches, elementos electrónicos, dispositivos móviles, etc. permanentemente conectados a Internet gracias a las nuevas tecnologías móviles como LTE y el diseño de IPv6 de estar permanentemente conectados a cualquier tipo de red sin ningún tipo de corte... **Y es que el futuro ya está aquí; es el presente.**

Lecturas recomendadas

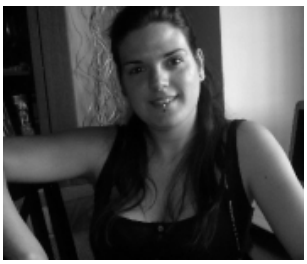
http://www.ipv6.es/es-ES/recursos/Paginas/Libros.aspx
http://www.worldipv6launch.org/
http://es.wikipedia.org/wiki/IPv6
http://www.ipv6.com/

Muchas gracias p4u (p4u@dabax.net) por la revisión del artículo y a xsf (www.xsf-coop.net) por la revisión idiomática ;)

Pásate a GNU/Linux con Arch: Gestores de ventanas y escritorios

Los usuarios que se inician en GNU/Linux deben tener claro los conceptos del entorno gráfico, para disponer de las herramientas suficientes a la hora de decidir qué utilizar.

Escrito por: **María José Montes Díaz** (Archera & Programadora)



Estudiante de Grado Ingeniería en Tecnología de la información. Técnico en informática de gestión. Monitora FPO. Docente de programación Python y Scratch para niños de 6-12 años. Activista del software libre y cultura libre.

Webs:
Blog: http://archninfa.blogspot.com.es/

Redes sociales:
Twitter: **@MMontesDiaz**

Una de las características de los sistemas GNU/Linux, es la posibilidad que nos brindan para adaptar nuestro sistema a nuestras necesidades y gustos. Para poder hacer esto, es conveniente aclarar determinados conceptos que, a veces, por su similitud, pueden hacer que un usuario recién iniciado confunda.

Una vez presentados estos conceptos, veremos dos escritorios desarrollados con la intención de aprovechar al máximo los recursos del equipo, sin necesidad de grandes requerimientos. Con ellos podremos aprovechar equipos antiguos, sin tener que recurrir a tecnologías obsoletas.

Gestor de ventanas o Window Manager

Es un conjunto de programas, ventanas y funcionalidades que hacen posible que el usuario pueda interactuar con el sistema de forma gráfica y no en modo texto. Antes de instalar un gestor de ventanas, es necesaria la instalación previa de un servidor X. Debemos tener claro que el gestor de ventanas y el servidor X son independientes.

El servidor X nos proporciona la base para poder disponer de interfaz gráfica. Actúa con arquitectura cliente-servidor. Es decir, es el servidor gráfico, sobre el que vamos a poder dibujar, mover ventanas, interactuar con el ratón y el teclado.

No es necesario en GNU/LINUX utilizar un servidor X y gestor de ventanas para usar el sistema. Estamos hablando de temas distintos e independientes entre si. Hay muchos usuarios que trabajan en modo texto, sin usar ninguna interfaz gráfica.

Existen muchos gestores de ventanas, el usuario es el que decide cual quiere utilizar, pudiendo tener más de uno instalado.

Podríamos distinguir tres tipos:

- **Stacking**: También conocidos cómo flotantes, son los utilizados, por ejemplo, en sistemas operativos comerciales y los más extendidos a nivel general. Se basan en entender la estación de trabajo cómo un escritorio tradicional, donde las aplicaciones serían los papeles que hay encima de él y podemos moverlos y colocarlos a nuestro antojo sobre el escritorio.
- **Tiling**: En este caso, la intención es la de no superponer las ventanas. Tienden a utilizar atajos de teclado y evitar, en mayor o menor medida, el uso del ratón. La gestión de ventanas se realiza en forma de mosaico y se puede configurar manualmente o utilizar los diseños predefinidos que nos suelen ofrecer.
- **Dynamic:** Permiten alternar de forma dinámica entre los dos diseños: mosaico y flotante.

Dentro del primer grupo, disponemos, entre otros, de:

- **WindowMaker:** se diseñó originalmente para el entorno de escritorio GNUstep. Es muy sencillo y potente, emula el entorno NeXT. Todavía sigue siendo utilizado por usuarios que les gusta su capacidad de personalización.
- **BlackBox:** Es un gestor de ventanas construido en C++ y con código completamente original, aunque la aplicación de gráficos se parece a la de WindowMaker. Su principal objetivo es aprovechar los recursos del sistema al máximo.
- **Fluxbox**. Basado en el código de BlackBox 0.61.1, es muy ligero y fácil de manejar. Dispone de una gran cantidad de opciones de personalización, lo que lo hace muy interesante.
- **OpenBox:** Es un gestor con un extenso soporte de estándares, altamente configurable. Su apariencia es minimalista. En sus inicios, se basó en BlackBox, pero fue reescrito completamente. Varios entornos de escritorio lo utilizan cómo gestor de ventanas, por ejemplo, LXDE o RAZOR-QT.
- **Compiz:** Es un gestor de composición de ventanas. El hecho de ser "de composición" hace referencia a que permite utilizar 2D y 3D a la hora de dibujar los elementos gráficos, proporcionando una gran cantidad de efectos visuales. Necesita, además, de un decorador de ventanas, como, por ejemplo, Emerald. No puede utilizarse junto a otros gestores, como Openbox, FluxBox o aquellos que incorporen composición.
- **Metacity:** Fue el gestor de ventanas utilizado por el proyecto Gnome. Presentaba una interfaz sencilla y amigable. Actualmente, ha sido sustituido por Mutter, aunque aún está disponible para los equipos que no poseen la potencia suficiente

para trabajar con Mutter.

- **Mutter:** Es un gestor de composición y ventanas para GNOME. Se basa en Clutter y utiliza OpenGL. Sobre este gestor, se pueden construir diferentes "Shell", para la gestión de las ventanas y el escritorio. El proyecto Gnome desarrolla Gnome-Shell, que es la interfaz sobre la que interactúa el usuario, programada en C y javascript, altamente configurable, con la posibilidad de utilización y creación de extensiones, que modifican y adaptan nuestro escritorio a nuestras necesidades.
- **Xfwm:** Es el gestor de ventanas del escritorio XFCE. Tiene su propio gestor de composición para ofrecer efectos visuales.
- **Kwin:** El gestor de ventanas estándar del escritorio KDE. También es un gestor de composición, similar a Compiz. Ofrece una gran cantidad de aspectos visuales configurables.

Del segundo grupo, cabe destacar:

- **Ion3:** Ion es un TWM diseñado pensando en el teclado de los usuarios. Fue el primero de este tipo de escritorios. La configuración de todo el gestor se realiza mediante el lenguaje de programación Lua.
- **Ratpoison:** Es simple, no contiene excesos y no posee grandes elementos gráficos. Se configura mediante un archivo de texto. En este gestor, el ratón carece de sentido.
- **Stumpwm:** El sucesor de Ratpoison. Está escrito en Common Lisp. No hay decoración de ventanas, ni botones, ni iconos. Tampoco dispone de bandeja de sistema.

En el tercer grupo, disponemos, entre otros, de:

- **Awesome:** Es un gestor altamente configurable, mediante Lua. Muy rápido y extensible. Dispone de bandeja de sistema y barra de información. Está desarrollado en C y Lua. Permite manejar las ventanas sin la necesidad del uso del ratón.
- **DWM:** Gestor de ventanas dinámico, con diseños aplicables de forma dinámica, que permiten la optimización del entorno. No dispone de archivos para configuración. Ésta se hace modificando el código fuente en C, por lo que es necesario recompilar cada vez que deseemos modificar algo.
- **WMII:** Este gestor, del mismo autor que DWM, es muy pequeño, rápido y con un consumo mínimo de recursos. Su interfaz utiliza el sistema de archivos 9P. El objetivo de este gestor es mantener un código base pequeño y limpio. Su configuración, por defecto, es en Bash, aunque también puede realizarse en Ruby.
- **Xmonad:** Este gestor está escrito y se configura con Haskell. Cualquier cambio que se quiera realizar en la configuración, debe volver a compilarse.

Por supuesto, existen muchos más gestores de ventanas en cada uno de los grupos, pero la intención no es mostrar todos los existentes, sino mostrar el amplio abanico de posibilidades que nos ofrece el hecho de utilizar un sistema GNU/Linux.

Entornos de Escritorio

Nos proporcionan los elementos para una interfaz gráfica de usuario, incorporando, entre otras cosas, iconos, ventanas, barras de herramientas, fondos de pantalla, además de aplicaciones y utilidades integradas, cómo gestores de archivos, navegadores web, etc.

En general, dentro de un escritorio, podríamos utilizar aplicaciones o utilidades pertenecientes a otro escritorio, brindándonos la posibilidad de adaptarlo a nuestros gustos y necesidades.

En el artículo anterior, tras instalar el sistema base, propuse la instalación de los dos grandes escritorios que hay en GNU/Linux: GNOME y KDE. Estos escritorios incluyen una gran cantidad de herramientas y, además, visualmente son muy atractivos. Sin embargo, requieren equipos con una potencia relativamente grande, aunque pueda optimizarse su consumo mediante la eliminación de efectos gráficos, o la utilización de aplicaciones más ligeras que las proporcionadas de manera estándar por ellos.

Ahora voy a centrarme en otros escritorios, disponibles en nuestra distribución, que están diseñados para ofrecer funcionalidades similares, pero con un costo muy inferior en recursos.

XFCE.- Este entorno de escritorio se escribió utilizando las herramientas GTK2. Dispone de aplicaciones cómo administrador de archivos (Thunar), administrador de ventanas (Xfwm) y paneles, entre otras.

Es más ligero que Gnome y KDE y su método de configuración es mediante GUI. Además, posee un compositor para permitir transparencias y aceleración. Trabaja bien con múltiples monitores.

El objetivo de este escritorio es ser rápido y con un consumo mínimo de recursos, sin dejar de lado el atractivo visual y la facilidad de uso.

Para instalarlo, disponemos de dos grupos de paquetes:

- **xfce4**: Incluye el escritorio, panel, sesión, herramientas de configuración, terminal, gestor de archivos y gestor de ventanas, entre otras cosas.
- **xfce4-goodies**: Aquí estarán los *plugins* para el panel, notificaciones y otras herramientas de sistema.

La instalación la realizaremos de la siguiente manera:

```
sudo pacman -Sy xfce4 xfce4-goodies gamin gvfs gvfs-afc udisks polkit-gnome
```

La configuración de Xfce puede realizarse de dos maneras:

1.- Mediante las opciones del Menú, de forma gráfica.

2.- Mediante una interfaz en línea de comandos, **xfconf-query**. Con esta herramienta, tenemos acceso a toda la configuración y permite realizar cambios en tiempo real. Veamos el uso de esta herramienta:

Cada elemento se le llama canal. Dentro de cada canal, disponemos de diferentes propiedades.

```
xfconf-query [opciones...]
```

Las opciones disponibles:

```
-h (help)        .- Mostrar opciones de ayuda.

-V (version)     .- Muestra la versión.

-c (channel)     .- El canal a consultar o modificar.

-p (property)    .- La propiedad a consultar o modificar.

-s (set)         .- Establece un nuevo valor a la propiedad.

-l (list)        .- Listar las propiedades (o canales, si -c no se
                    especifica).

-v (verbose)     .- Salida detallada.

-n (create)      .- Crear una nueva propiedad, si ésta no existe.

-t (type)        .- Especificar el tipo de  valor de la propiedad.

-r (reset)       .- Restablecer propiedad .

-R (recursive)   .- Recursivo (usar con -r)

-a (force-array).- Forzar como vector incluso si sólo hay un elemento.

-T (toggle)      .- Invertir una propiedad booleana existente

-m (monitor)     .- Monitorizar un canal.
```

Algunos ejemplos:

Para listar todos los canales disponibles:

```
xfconf-query -l
```

Para listar las propiedades del canal xfce4-panel:

```
xfconf-query -c xfce4 -l
```

Para listar las propiedades del canal xfce4-panel y sus sus valores:

```
xfconf-query -c xfce4-panel -l -v
```

Para activar la composición:

```
xfconf-query -c xfwm4 -p /general/use_compositing -s true
```

O para invertir el valor:

```
xfconf-query -c xfwm4 -p /general/use_compositing -T
```

En principio, los requisitos mínimos para este escritorio son:

- Procesador: 300Mhz.
- Memoria: 128 MB.
- Disco Duro: 90 MB.

LXDE.- Es el acrónimo en inglés de "Lightweight X11 Desktop Environment" (entorno de escritorio ligero para X11) y, además, LX puede referirse a LinuX.

Los componentes de este escritorio tienen escasas dependencias entre ellos, por lo que pueden utilizarse individualmente. Son independientes.

El proyecto LXDE intenta ofrecer un nuevo entorno de escritorio que sea suficientemente útil, manteniendo a la vez un uso reducido de recursos. La usabilidad, la rapidez y la utilización de memoria son las prioridades de los desarrolladores.

Esta compuesto por:

- **PCManFM:** Gestor de ficheros, proporciona además los iconos del escritorio

- **LXPanel:** Panel de escritorio.
- **LXSession:** Gestor de sesiones ajustado a los estándares con funciones de apagado, reiniciado y suspensión.
- **LXAppearance:** LXAppearance es un seleccionador de temas GTK+ muy completo, capaz de cambiar temas GTK+, temas de iconos e incluso las fuentes tipográficas usadas por las aplicaciones
- **Lxappearance-obconf:** Es un plugin para Lxappereance que permite configurar OpenBox.
- **Gestor de ventanas:** Openbox, icewm, fluxbox, metacity, etc. Por defecto, LXDE utiliza Openbox.
- **GPicView:** Un visualizador de imágenes sencillo, rápido y ligero que arranca al instante.
- **Leafpad:** Editor de texto ligero y sencillo: No forma parte del proyecto LXDE, pero se recomienda su uso.
- **XArchiver:** Programa archivador ligero, rápido, basado en gtk+ e independiente del escritorio. Igual que ocurre con Leafpad, no forma parte del proyecto, pero es el editor que se recomienda.
- **LXNM :** Gestor de red ligero para LXDE con capacidad de conexión sin hilos (sólo para Linux). Su desarrollo está parado, por lo que es aconsejable utilizar otro gestor de red.
- **Lxtask:** Un gestor de tareas ligero.
- **Menu-cache:** Un demonio (*daemon*) encargado de generar el menú de aplicaciones.

La instalación la realizaremos de la siguiente manera:

```
# pacman -Sy lxde leafpad obconf epdfview polkit-gnome gvfs gvfs-afc upower
```

Para tener acceso a la papelera en el administrador de archivos:

```
mkdir -p ~/.config/autostart

cp /etc/xdg/autostart/polkit-gnome-authentication-agent-1.desktop
~/.config/autostart
```

Ahora, editamos el archivo

`~/.config/autostart/polkit-gnome-authentication-agent-1.desktop`

y añadimos LXDE a la siguiente línea:

```
OnlyShowIn=GNOME;XFCE;LXDE;
```

Para reemplazar el gestor de ventanas openbox por otro, por ejemplo, compiz, editamos el archivo `/etc/xdg/lxsession/LXDE/destktop.conf` y modificamos la opción `window_manager`:

```
[Session]
window_manager=compiz ccp --indirect-rendering
```

Los requisitos mínimos para LXDE son:

- Procesador: Pentium II 266 Mhz
- Memoria: Aunque, una vez arrancado LXDE, el uso total de memoria es de 45 MB, es aconsejable disponer de, al menos, 128 MB
- Disco: 60 MB

Iniciando sesión

Para iniciar sesión, por defecto Arch arranca en modo multiusuario, que es en modo texto. Podemos utilizar gestores de login gráfico para el inicio de sesión. Primero debemos activar el arranque en modo gráfico. Lo haremos así:

```
sudo rm /etc/systemd/system/default.target
sudo systemctl enable graphical.target
```

Desde el modo multiusuario también es posible, por supuesto, lanzar nuestro escritorio preferido.

Disponemos de varios gestores de login gráficos. Veamos los más usuales:

GDM.- Es un gestor de login perteneciente al proyecto Gnome. Para instalarlo:

```
sudo pacman -S gdm
```

KDM.- Es el gestor de login del proyecto KDE. Lo instalaremos así:

```
sudo pacman -S  kdebase-workspace
```

SLiM.- Acrónimo de “Simple Login Manager”. Es un gestor de login muy liviano. Su configuración para la elección de sesión se realiza mediante la edición del archivo ~/.xinitrc.

```
sudo pacman -S slim
```

Si optamos por KDM y pretendemos lanzar LXDE, deberemos crear una entrada para que KDM lo reconozca y nos permita seleccionarlo:

```
sudo cp /usr/share/xsessions/LXDE.desktop /usr/share/apps/kdm/sessions/
```

Tanto para la utilización de SLiM, cómo para iniciar sesión en un escritorio desde consola, debemos crear el archivo de configuración ~/.xinitrc:

```
cp /etc/skel/.xinitrc ~
chmod +x ~/.xinitrc
```

Ahora, al final del archivo, añadimos, según sea el escritorio que decidamos instalar, uno de los siguientes comandos:

```
exec gnome-session
exec startkde
exec startlxde
exec startxfce4
```

Además, si pretendemos lanzar con SLiM el escritorio Gnome o vamos a utilizar herramientas de este escritorio, habrá que editar el archivo `/etc/pam.d/slim` y añadir:

```
auth       optional    pam_gnome_keyring.so
session    optional    pam_gnome_keyring.so    auto_start
```

En el archivo `/etc/pam.d/passwd` también debemos añadir:

```
password        optional        pam_gnome_keyring.so
auth            optional        pam_gnome_keyring.so
```

Ahora, ya tenemos nuestro gestor de login instalado. Además, tenemos habilitado el arranque en modo gráfico. Sólo falta habilitar el gestor correspondiente. Debemos tener en cuenta que sólo podemos iniciar un gestor a la vez.

Habilitar en el inicio un gestor:

```
sudo systemctl enable gdm|kdm|slim
```

Para deshabilitarlo:

```
sudo systemctl disable gdm|kdm|slim
```

En el caso de no querer utilizar login gráfico, desde la terminal, sólo debemos ejecutar `startx` y se lanzará el escritorio que hayamos especificado en el `~/.xinitrc`.

No sólo es posible iniciar sesión gráfica con un escritorio completo. También sería posible hacerlo con, únicamente, un gestor de ventanas, por ejemplo. Además, podemos tener los gestores y escritorios que deseemos instalados a la vez, e ir utilizando el que nos convenga en cada momento.

Enlaces de interés:

http://lxde.org/

http://www.xfce.org/

http://slim.berlios.de/

https://wiki.archlinux.org/index.php/Window_manager

https://wiki.archlinux.org/index.php/Desktop_Environment

Invitación al proyecto GcalcTool: GNOME Calculator

GNOME no solo brinda un entorno de escritorio genial: también cuenta con muchas aplicaciones muy útiles para usuarios y programadores, las cuales muchas personas ni imaginan las funcionalidades con las que cuentan. Por ejemplo Calculator, una simple pero poderosa herramienta.

Escrito por: **Milagros Alessandra Infante Montero** (Est. Ing. Informática)

Estudiante de Ingeniería Informática. Miembro de **APESOL** (Asociación Peruana de Software Libre) y de la comunidad de software libre **Lumenhack**. Miembro del equipo de traducción al español de **GNOME**. Apasionada por el desarrollo de software, tecnología y gadgets. Defensora de tecnologías basadas en software libre y de código abierto.

Webs:
Blog: www.milale.net

Redes sociales:
Twitter / Identi.ca: **@milale**

Calculator[34] es una aplicación de GNOME que provee cálculos matemáticos de nivel simple y medio, con eficiencia y es conveniente como una aplicación predeterminada en un entorno de escritorio.

La calculadora ya viene de manera predeterminada con el entorno de escritorio de GNOME pero también puede ser usada en otras distribuciones, ya que la herramienta provee más funcionalidades que las que parece ofrecer por defecto.

Para la instalación en cualquier distribución que soporte `apt-get`, solo se hace uso del siguiente comando:

```
sudo apt-get install gcalctool
```

34 https://live.gnome.org/Calculator

Características

- Es una herramienta para cálculos de ecuaciones matemáticas.
- Usa notación matemática estándar donde sea posible (los usuarios no tienen que aprender la aplicación al ya saber matemática).
- Es lo suficientemente simple para matemática básica (sumar, restar, dividir, multiplicar).
- Es lo suficientemente poderosa para matemática de nivel medio.
- Fácil de cargar y de responder a las entradas.
- Tiene el tamaño apropiado para encajar en resoluciones de pantalla estándares (por ejemplo, Netbooks).
- Totalmente accesible.
- Bien documentado.

Pero también es importante saber las características con las que no cuenta:

- No emula ninguna interfaz de calculadora existente, ni hardware, ni software.
- No es una herramienta poderosa para matemáticos profesionales (los profesionales deben usar una aplicación especial).
- No es un lenguaje de programación.

Interfaz de la calculadora de GNOME GcalcTool

La interfaz simple con la que cuenta a primera vista, se cambia de acuerdo al modo en el que lo estés usando.

- **Barra de menús:**

  Está ubicada en la parte superior. Aquí se encuentran todas las acciones y opciones disponibles para el GcalcTool.

- **Barra de visualización:**

  Los resultados calculados se muestran en esta barra y los números también se muestran al escribir o pulsarlos.

- **Área de modo:**

  Es donde todos los botones están mostrados, los botones disponibles cambiarán de acuerdo al modo elegido.

- **Barra de estado:**

  Se mostrará el estado de sus cálculos.

Ecuaciones de la calculadora de GNOME GcalcTool

GcalcTool es capaz de trabajar con muchos tipos de ecuaciones, las más comunes son:.

- Aritmética básica
- Trigonometría
- Potencia
- Raíz cuadrada

> *"La matemática parece dotar a uno, de un nuevo sentido".*
> *Charles Darwin*

Tamaño de la palabra

Un detalle importante a tomar en cuenta está dentro de las Preferencias de Calculator. Nosotros podemos elegir el tamaño de la palabra (la cadena finita de bits) ya que al momento de diseñar una arquitectura de computadores, este dato es fundamental. Calculator soporta palabras de 8 a 64 bits.

Modos de la calculadora de GNOME GcalcTool

Los modos ofrecidos por esta herramienta se dan si se necesita de funcionalidades extras.

- **Modo básico:**

  Este es el modo predeterminado que provee las funcionalidades simples y cálculos básicos.

- **Modo avanzado:**

  Revela naturalmente todas las funcionalidades avanzadas.

- **Modo financiero:**

  Este modo lo ayudará con sus cálculos financieros. Ideal para testear cálculos financieros y asegurarse de ser los correctos, antes de "codearlos".

- **Modo científico:**

  Los usuarios pueden probar este modo para cálculos trigonométricos.

- **Modo programador:**

  Este modo provee funcionalidades adicionales que los programadores pueden encontrar útiles y es en este modo, en el cuál nos enfocaremos.

Modo programador de GcalcTool

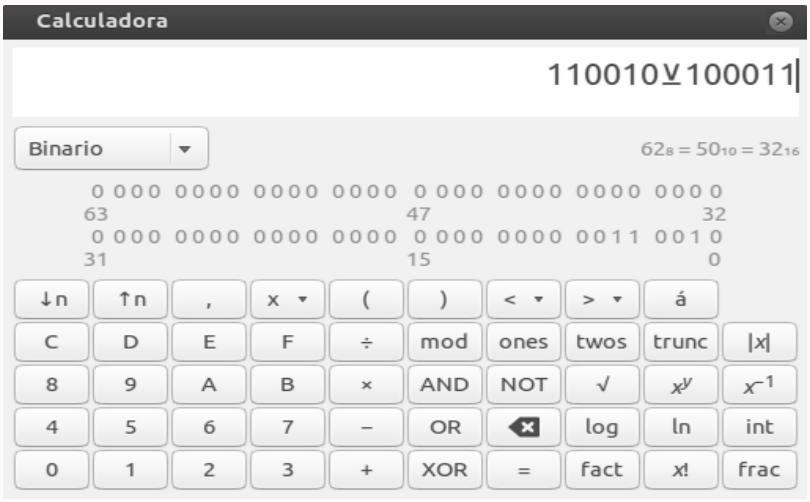
Como se menciona en la sección de arriba sobre los modos de Calculator, existe un modo programador poco conocido pero el cual proporciona ecuaciones útiles como funciones bit a bit y un editor de los mismos.

Veamos algunos ejemplos:

- **Algebra Booleana:**

  Calculator soporta las funciones `AND`, `OR`, `XOR`, `NOT`.

  `110010 XOR 100011` nos dará como resultado `10001`



- La función `NOT` invierte los bits en un número. Con `NOT 110100` obtendremos `001011`

- **Códigos de caracteres:**

  El botón **á** convierte caracteres en códigos de caracteres. Por ejemplo el símbolo ~ en hexadecimal equivale a `7E`

¿Como colaborar con Calculator?

Los errores y requerimientos de características deben ser presentadas en el Sistema de Seguimiento de Errores de GNOME[35], rellenando la información del error que hallaste[36]. Y si quieres ser parte del proyecto e involucrarte en él para su mejora, puedes inscribirte en la lista de correo[37]. Al leer un poco en la Ayuda de GcalcTool podremos verificar algunas funcionalidades con las que aún no cuenta, como el soporte a números complejos o el soporte a funciones definidas por el usuario. **No dudes en participar de la lista de correos y quizás llegar a involucrarte en darle un "improve" a Calculator**.

35 https://bugzilla.gnome.org/
36 https://bugzilla.gnome.org/enter_bug.cgi?product=gcalctool
37 https://mail.gnome.org/mailman/listinfo/gcalctool-list

Ubuntu Rookie: Toma 3

Por **Eliana Caraballo** (@elianaca)

Ingeniera de desarrollo senior y miembro activo de AVANET. Apasionada por todo lo relacionado con la ingeniería de software y los procesos de calidad para desarrollo. Siempre abierta a aprender y a compartir lo que los años de desarrollo me han enseñado.
http://co.linkedin.com/in/elianacaraballoa

La historia comenzó con un: "No eres tú Ubuntu, soy yo quien de verdad no te entiende" y tres intentos después, por fin pude adaptarme a este sistema operativo. Esta es mi historia...

Estaba en los últimos semestres de la universidad cuando escuché sobre un sistema operativo que podía correr desde un CD y no había necesidad de instalarlo. Por esos días necesitaba recuperar cierta información de mi equipo que ya estaba listo para ser formateado porque el SO que tenía no quería arrancar -ni siquiera en modo a prueba de fallos- así que decidí probar cómo funcionaba mientras realizaba el *backup* de mi información. Pero cuando llegó el momento de configurar la red wi-fi me di cuenta que no era tan sencillo como estaba acostumbrada, así que preferí aplazar la prueba para después.

Mi segundo intento con **Ubuntu** vino unos 3 años después, cuando escuché a un amigo hablar acerca de *compiz* y que habían mejorado su interfaz gráfica. Así que de nuevo lo descargué y descubrí con real agrado que la configuración de la red inalámbrica era muchísimo más sencilla y que visualmente era todo más atractivo. En esa oportunidad, el intento falló porque estaba realizando algunos trabajos por mi cuenta y el software que usaba no era compatible con el sistema operativo, ni siquiera usando **Wine** y el manejo de la máquina virtual se me hizo complicado para los propósitos que tenía en ese momento.

Comentando mi problema con algunos colegas, se generaron discusiones sobre las diferencias y ventajas entre *X* e *Y* sistemas operativos. LLegamos a la conclusión de que **Ubuntu** era tan técnico, que se hacía demasiado complicado para un usuario no técnico o con pocos conocimientos al respecto. Por lo menos a mis ojos se hizo patente la necesidad de generar programas "amigables" porque al final, no vamos a ser nosotros quienes interactuaremos con el software que realizamos en nuestro día a día, sino que siempre hay que pensar en el usuario final que tendrá que "lidiar" con nuestro sistema.

Si no nos esforzamos para que su configuración y manejo sean amigables, muy probablemente nuestro cliente se sentirá frustrado y preferirá buscar otra alternativa donde se sienta más cómodo.

He descubierto que la mayoría de los programadores, hacemos todo tan obvio a nuestros ojos de ingenieros de sistemas que nos olvidamos del perfil de las personas que realmente van a usar el software y podemos "complicar" su usabilidad.

La amigabilidad comenzó como un hito en 1998 al comenzar a aproximar al usuario con el software a través de tareas sencillas como la configuración, restauración y demás necesidades que tuviera frente a su información. Se tenía una interfaz gráfica que interactuaba con el usuario con la intención de facilitar la satisfacción de sus necesidades y una

serie de procesos independientes automáticos, encargados de las tareas que no necesitaban ser monitorizadas.

A partir de ese momento, casi todo el software que se produce usa este mismo modelo donde el usuario simplemente interactúa con pantallas que le permiten alimentar un sistema que tiene la capacidad de realizar las tareas que necesita.

La "amigabilidad" de un sistema permite que se vuelva intuitivo para el usuario final y que su facilidad de uso aumente la satisfacción de nuestro cliente, quien es al final para el que estamos trabajando.

Poco o nada importa en qué lenguaje haya sido escrito; lo que importa es que para ellos, no se vuelva una tortura de parches, comandos y *truquios* para que funcione como se supone debería funcionar. La parte "pesada" debe estar siempre en la complejidad de las reglas de negocio y nuestra interfaz, lo más intuitiva, sencilla y amigable posible.

En mi opinión, fue esa falta de amigabilidad en **Ubuntu** en los dos primeros intentos, lo que hizo que no me enamorara del todo del sistema operativo, porque al pensar en todos los que en ese momento usábamos el PC en la casa, comprendí que no iba a ser fácil que se adaptaran al cambio.

Tal parece que **Ubuntu** se dio cuenta y tomó nota de estas cosas, porque **la versión que actualmente tengo instalada ha sido bastante interesante:** Hay una tienda donde el software se descarga solo y facilita el trabajo y a su vez sirve de manejador, si hay que descargarlo desde la web; la suite de LibreOffice con la que viene instalada es fabulosa y tiene bastantes utilidades; la configuración del sistema es sencilla e intuitiva.

Podrás pensar que soy toda una *Rookie* emocionada con mis "nuevos descubrimientos", pero para ser el tercer intento con el sistema operativo instalado en mi PC, debo reconocer que ha mejorado muchísimo, pues se hizo mucho más sencillo manejarlo y es mucho más amigable con usuarios que no son técnicos en computación. Todavía hay cosas por mejorar en cuanto a compatibilidad con algunos programas, pero en conjunto, ha sido una experiencia muchísimo más agradable y varios meses después, todavía no he sentido las ganas de volver a mi antiguo sistema operativo.

> *"Recuerda: no eres torpe, no importa lo que digan esos libros. Los torpes de verdad son gente que, creyéndose expertos técnicos, no podrían diseñar hardware y software manejable por usuarios normales, aunque la vida les fuera en ello." Walter Mossberg – Columnista de The Wall Street Journal*

"Elvis"

Hackers & Developers Magazine

ASCII Art

U! Tu zona exclusiva

Para publicar tu mensaje en la Zona U!, envíanos un e-mail contacto@hdmagazine.org, indicando ZonaU! en el asunto del mensaje.

HDMagazine Responde:

Matías Caricato (Argentina):
(...) Primero que nada, quiero felicitar a todos los que participan en la creación de esta excelente revista que llena nuestros espacios de ocio aportando temas de lectura muy interesantes y de primera calidad. Mi sugerencia es la siguiente: agregar a la web un módulo de subscripción que notifique por email cuando hay una nueva edición de la revista disponible (...)

Aura María Acosta (Colombia):
(...) Quiero saber como hago para suscribirme a esta revista, no perderme un solo numero ya que mi campo de acción son los sistemas; reparo computadores y dicto clases.

Respuesta a Aura y Matías: *Hola Aura y Matías! Muchas gracias por escribirnos. Para poder ofreceros un sistema de suscripción por e-mail, necesitaríamos contar con un servidor de correo y personal que se encargue de mantenerlo. Confiamos en que los aportes voluntarios que puedan hacer nuestros lectores, en un futuro, nos podrán ayudar a solventar los costes que este servicio implicaría.*

Richard Diaz Rodriguez (Cuba):
Me gustaría que hicieran alusión a la seguridad de las comunicaciones GSM y sobre si es cierto determinar o triangular la posición de un comunicación móvil o vía GPS. Saludos a todos(as) creo que están haciendo un trabajo magnifico y por acá por cuba se les sigue de muy cerca y se cree y quiere mucho su trabajo. Feliz 2013.

Respuesta: *Hola Richard! Muchas gracias por tu mensaje! Seguro que* ***HackLadino*** *cumplirá tu deseo!*

Lailah (Uruguay):
Hola a todos! Leí la última edición con la entrevista a Richard Stallman. ¿Podrían en algún momento incluir una entrevista a Linus Torvalds? Creo que sería muy interesante.

Respuesta: *Hola Lailah! Promesa que intentaremos hacer todo lo posible por cumplir tu deseo en 2013 =)*

Fabian Arellano (Chile):
hola quiero agradecerles por la revista es muy instructiva e informativa y si en el otro numero de su revista puedan incluir un apartado de inteligencia artificial aplicado a Juegos Saludos y feliz año mi nombre es Fabián y soy de Chile

Respuesta: *Hola Fabian! Muchas gracias por tu mensaje! Por el momento, no contamos con columnistas especializados en el desarrollo de videojuegos, pero sí, con una experta en IA (Yecely). Veremos que puede hacer nuestra experta al respecto.*

Stefan (España):
Me encanta vuestro trabajooo!! grande... muy grande!! Un saludo desde España!!

Respuesta: *Hola Stefan! Muchísimas gracias por tu mensaje! Un placer leerte :)*

Anuncios:

FIREFOX OS APP DAYS

El pasado sábado 26 de enero de 2013, se lanzaron los Firefox OS App Days en diferentes ciudades de todo el mundo. Fuimos invitadas/os a cubrir el evento y nuestra columnista **Laura Mora estuvo en el Firefox OS App Day Barcelona.** En la próxima edición, tendremos los detalles en exclusiva. Más información sobre Firefox OS: www.mozilla.org/es-ES/firefoxos/

LANZAMIENTO DE OPENWEBINARS

La **Fundación Guadalux** los invita a participar de los nuevos seminarios Web gratuitos sobre aplicaciones, herramientas y tecnologías para el desarrollo de proyectos OpenSource, que pueden aplicarse al entorno empresarial. Más información: www.guadalux.org/openwebinars

¡GRACIAS A TOD@S POR LEERNOS!