AÑO 0
NÚMERO 0
2012-11-05

#0

<Butterfly>

HD Hackers & DEVELOPERS

Magazine digital de distribución mensual
sobre Software Libre, Hacking
y Programación

Staff

Celia Cintas Licenciada en Informática
Eugenia Bahit Arquitecta GLAMP & Agile Coach
Eliana Caraballo Ingeniera de Sistemas
Filly Programadora
Indira Burga Ingeniera de Sistemas
Milagros Infante Est. Ingeniería Informática
Sorey García Arquitecta de Software
Yecely Díaz Maestra en Inteligencia Artificial

HD Hackers & DEVELOPERS #0

Acerca de

Hackers & Developers es un Magazine digital de distribución libre y gratuita, sobre Software Libre, hacking y programación.

Se distribuye mensualmente bajo una licencia Creative Commons.

Colabora

¿Quieres colaborar con HD Magazine? ¡Hay varias formas de hacerlo!

Envía tu artículo
Puedes enviarnos tu artículo a **colabora@hdmagazine.org**

Distribuye la revista
Ayúdanos a llegar a más programadoras/es distribuyendo la revista a tus contactos o publicando enlaces a **www.hdmagazine.org** en las redes sociales.

Haz un donativo
Puedes apoyar a HD Magazine Económicamente.
Envíanos tu propuesta a:
donaciones@hdmagazine.org

Contacta

Si lo deseas, puedes enviar tu **consulta** a **contacto@hdmagazine.org** para que una de nuestras expertas, te responda en el siguiente número.

También puedes enviarnos tus **comentarios**, sugerencias u opiniones sobre la revista, escribiéndonos a **lectores@hdmagazine.org**

Este mes en Hackers & Developers...

"... Hacker es alguien que disfruta jugando con la inteligencia..."

Richard Stallman

Free Software, Free Society
Pág. 97, GNU Press 2010-2012

ASCII ART
Este mes: «Butterfly»
by **chris.com**

>> Pág. 48

Y ahora ¿qué Framework PHP usaré?

PHP

Hace muchos años tomé la decisión de especializarme en el desarrollo de Software web; me volví «free» -específicamente con el lenguaje PHP-. Por supuesto, inicié los «pininos» con la programación estructurada -como la mayoría-. Mientras iba desarrollando más aplicaciones me daba cuenta de que trabajaba de manera desordenada y que existía mucha redundancia de código, por supuesto ahí conocí POO y por ende qué es un Framework (1) y, casi instantáneamente vi la luz.

Escrito por: **Indira Burga** (Ingeniera de Sistemas)

Indira es **Ing. de Sistemas** de Perú. Gestora de Proyectos de desarrollo de software, **programadora PHP**, analista, nueva amante de las **metodologías Ágiles**. Ahora envuelta en una nueva aventura: su propia empresa "**IC Projects**" dedicada al desarrollo de Software.

Webs:
About.me: http://about.me/indirabm

Redes sociales:
Twitter: **@indirabm**

La mayoría de los programadores prefieren trabajar con un framework, una de las más grandes razones es porque nos facilita el desarrollo de software, evitando reescribir el mismo script, además de contar con el soporte de una comunidad (2) que está en constante movimiento mejorando y creando nuevas librerías que valgan verdades. Es un alivio tenerlas a la hora de desarrollar. También hay que recordar que todos ellos utilizan las mejores prácticas, por tanto podremos centrar los esfuerzos en los requerimientos de cliente.

Primero empezaremos mencionando que existen muchos frameworks de PHP pero que

para elegir algunos hay que tener en cuenta ciertos criterios, los mencionare a continuación:

- **Rapidez / performance**, lo que permite al usuario una mejor experiencia en la web.
- **Separación de vista y código PHP**, cuando se trabaja un proyecto trabajamos con dos tipos de equipos, los desarrolladores y los diseñadores, los dos trabajan al mismo tiempo por lo tanto el framework nos tiene que brindar esa facilidad.
- **Seguridad**, esto permitirá contar con los mínimos requisitos de protección contra ataques XSS y CSRF.
- **Soporte de versiones**, que soporte la última versión de PHP.
- **ORM**(3), Contar con una propia o en su defecto que soporte otras.
- **Popularidad y tamaño de la comunidad**, lo que más se reconoce de un framework es el crecimiento que ha tenido, las nuevas ideas, la cantidad y calidad de plugins, etc.
- **Curva de aprendizaje**(4), recomendable para poder mejorar tu nivel de programación.

Un ejemplo común para ver la diferencia entre POO y programación estructurada es: si quieres validar un e-mail que recibes de un formulario, en vez de tener que implementar toda esa parte (y además hacerlo bien) puedes hacer uso de clases existentes en Zend como `Zend_Validate_EmailAddress()`, y lo mismo pasa en los demás frameworks:

Programación estructurada

```
function comprobar_email($email){
   $mail_correcto = 0;
   //compruebo unas cosas primeras
   if ((strlen($email) >= 6) && (substr_count($email,"@") == 1) &&
       (substr($email,0,1) != "@") &&
       (substr($email,strlen($email)-1,1) != "@")){
       if ((!strstr($email,"'")) && (!strstr($email,"\"")) &&
           (!strstr($email,"\\")) && (!strstr($email,"\$")) &&
           (!strstr($email," "))) {
       //miro si tiene caracter .
       if (substr_count($email,".")>= 1){
       //obtengo la terminacion del dominio
       $term_dom = substr(strrchr ($email, '.'),1);
       //compruebo que la terminación del dominio sea correcta
       if (strlen($term_dom)>1 && strlen($term_dom)<5 &&
          (!strstr($term_dom,"@")) ){
            //compruebo que lo de antes del dominio sea correcto
            $antes_dom = substr($email,0,strlen($email) - strlen($term_dom) - 1);
            $caracter_ult = substr($antes_dom,strlen($antes_dom)-1,1);
            if ($caracter_ult != "@" && $caracter_ult != "."){
             $mail_correcto = 1;
            }
```

```
            }
          }
        }
      }
      if ($mail_correcto)
          return 1;
      else
          return 0;
  }
```

POO

```
$email = $this->getRequest()->getPost('email', 'none@example.com');
$validator = new Zend_Validate_EmailAddress();
```

¿Que hay en el menú del día?

Hoy por hoy existen -como lo mencionamos anteriormente- muchos frameworks de PHP, por cierto algunos podrían ser mejor para ciertos tipos de desarrollo, la mayoría de ellos cumplen con muchos de los criterios que mencionamos, pero listare los que a mi parecer son los mejores:

Zend Framework: Este framework es robusto con muchas librerías y que claro tiene como respaldo a Zend la empresa creadora de PHP, lo cual le da mucha más solvencia, algo que tiene Zend que no los demás es el desacoplamiento (5) cuenta con una línea de aprendizaje alta, esto a mi parecer es muy importante, por cierto es el framework que actualmente uso.

Url: http://framework.zend.com/

Symfony: Este es otro de los frameworks más sólidos, cuenta con miles de bundles (plugins) [o librerias para Zend], de hecho se toma muy en serio todo lo referente a seguridad, tanto que incluso han pagado una cara auditoría del código llevada a cabo por una empresa experta en seguridad, además cuenta con una ORM muy buena que es Doctrine.

Url: http://symfony.com/

CakePHP: Fue el primer framework que use, aunque por poco tiempo, es sencillo de ahí su nombre, buena documentación, considero que su curva de aprendizaje es menor lo cual también da una buena salida para proyectos pequeños.

Url: http://cakephp.org/

Yii: Este es otro de mis favoritos lo recomiendo para aquellos que recién se incursionan

en un framework rápido de aprender, y cuenta con un generador de archivos vía web (llamado Gii), que permite crear : Módulos, controladores, modelos y formularios CRUD (6) , que te hace la vida fácil. Realmente puedes tener aplicaciones listas en minutos y con una vista respetable. Yii también cuenta con muchos plugins.

Url: http://www.yiiframework.com/

Recomiendo que visiten esta página:

http://www.bestwebframeworks.com/compare-web-frameworks/php/

Una pregunta muy interesante que yo misma me hice es, **¿Y podría desarrollar mi propio framework?** , pues la respuesta es un SI rotundo, por supuesto que como regla es usar la mejores prácticas, recuerden que otras personas podrían darle mantenimiento en el futuro.

Notas:

(1) Un Framework es (plataforma, entorno, marco de trabajo). Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

(2) La comunidad del software libre es un término que hace referencia informal a los usuarios y desarrolladores de software libre, así como los partidarios del movimiento de software libre.

(3) En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo), facilitando el trabajo con diferentes base de datos.

(4) Una curva de aprendizaje describe el grado de éxito obtenido durante el aprendizaje en el transcurso del tiempo. A menudo se cometen muchos errores al comenzar una nueva tarea. En las fases posteriores disminuyen los errores, pero también las materias nuevas aprendidas

(5) Se pueden trabajar las librerías por separado y si se desea utilizarlas en otro framework.

(6) CRUD es el acrónimo de las cuatro funciones básicas de acceso a una base de datos (Crear, Obtener, Actualizar y Borrar)

Creando una capa de abstracción con PHP y mysqli

PHP

mysqli, es el conector para bases de datos MySQL recomendado por PHP, para interactuar desde tu aplicación con una base de datos MySQL. Pero crear una capa de abstracción genérica, reutilizable y orientada a objetos, suele ser un dolor de cabeza. En este artículo, veremos como lograr crear una capa de abstracción a bases de datos, simple y muy fácil de usar.

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
Agile Coaching: www.eugeniabahit.com

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

Si alguna vez intentaste crear una capa de abstracción a bases de datos, utilizando el conector mysqli en estilo orientado a objetos, sabrás que hacerlo es un gran dolor de cabeza.

Pero **¿qué es exactamente mysqli y en qué se diferencia de mysql?** Básicamente, como bien se define en el manual oficial de PHP, mysqli es **"una extensión mejorada del conector mysql"**. Entre las principales diferencias, se encuentran además, sus dos grandes ventajas:

- Permite trabajar en estilo orientado a objetos (también continúa proveyendo soporte para estilo procedimental);

- Nos facilita una forma segura de filtrado de datos en sentencias SQL, para prevenir inyecciones SQL;

Sin dudas, mysqli es una gran ventaja frente al antiguo conector. Tiene una gran cantidad de clases, métodos, constantes y propiedades muy bien documentados[1]. Sin embargo, entender la documentación puede ser una tediosa tarea, en la cual, hallar un principio y un fin, se podrá convertir en la peor pesadilla de tu vida.

Así que entonces ¡Manos a la obra! Y a crear una capa de abstracción con mysqli orientado a objetos.

Recetario para crear una capa de abstracción a bases de datos con mysqli

Lo primero que debemos tener en cuenta, es que nuestra capa de abstracción deberá proveer de métodos públicos, que puedan ser llamados de forma estática, para crear un objeto conector, no sea necesario.

Para poder lograr una capa de abstracción genérica, **la clave** es utilizar **`ReflectionClass`**[2] para crear una instancia de **`mysqli_stmt`** y mediante `ReflectionClass->getMethod`, invocar al método **`bind_param`**. De lo contrario, preparar una consulta SQL y enlazarle los valores a ser filtrados, será imposible.

> *Ten en cuenta que para seguir los ejemplos de este artículo, es necesario contar con la **versión 5.3.6** o superior, de **PHP**.*

Propiedades

Nuestra capa de abstracción, tendrá una única propiedad pública, destinada a almacenar los datos obtenidos tras una consulta de selección. El resto de las propiedades, serán de ámbito protegido, accesibles solo desde nuestra clase y clases que hereden de ésta.

```
class DBConnector {

    protected static $conn;        # Objeto conector mysqli
    protected static $stmt;        # preparación de la consulta SQL
    protected static $reflection;  # Objeto Reflexivo de mysqli_stmt
    protected static $sql;         # Sentencia SQL a ser preparada
    protected static $data;        # Array conteniendo los tipo de datos
                                     más los datos a ser enlazados (será
                                     recibido como parámetro)
```

1 http://php.net/manual/es/book.mysqli.php
2 http://php.net/manual/es/class.reflectionclass.php

```
    public static $results;        # Colección de datos retornados por una
                                     consulta de selección

}
```

La consulta SQL, deberá ser seteada en los modelos (clases) donde se requiera, incluyendo marcadores de parámetros (embebidos con el signo ?), en la posición donde un dato deba ser enlazado. Un ejemplo de ella, podría ser el siguiente:

```
$sql = "INSERT INTO productos (categoria, nombre, precio)
        VALUES (?, ?, ?)";
```

Mientras que el array `$data`, deberá contener, como primer elemento, una string con el tipo de datos y los elementos siguientes, serán los datos a ser enlazados (todos de tipo *string*):

```
$data = array("isbd",
              "{$categoria}", "{$nombre}", "{$descripcion}", "{$precio}");
```

El primer elemento, siempre representará el tipo de datos correspondiente al marcador de parámetro que se desea enlazar. Siendo los tipos de datos posibles: ***s*** (string), ***i*** (entero), ***d*** (doble) y ***b*** (blob).

Métodos

Conectar a la base de datos:

```
    protected static function conectar() {
        self::$conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
    }
```

Requerirá 4 constantes predefinidas (se recomienda definir en un archivo `settings`): `DB_HOST`, `DB_USER`, `DB_PASS` y `DB_NAME`.

Preparar una sentencia SQL (con marcadores de parámetros):

```
    protected static function preparar() {
        self::$stmt = self::$conn->prepare(self::$sql);
        self::$reflection = new ReflectionClass('mysqli_stmt');
    }
```

La clase `ReflectionClass` de PHP, cumple un papel fundamental: solo a través de ella podemos crear un objeto `mysqli_stmt` reflexivo, siendo ésta, la única alternativa que tenemos para preparar sentencias SQL con marcadores de parámetros, de forma dinámica.

La propiedad estática `$sql` (`self::$sql`) será creada por el único método público que tendrá la clase.

Enlazar los datos con la sentencia SQL preparada:

```
protected static function set_params() {
    $method = self::$reflection->getMethod('bind_param');
    $method->invokeArgs(self::$stmt, self::$data);
}
```

La propiedad estática `$data` que se pasa como segundo parámetro a invokeArgs, también será seteada por el único método público.

En este método (`set_params`), la variable temporal `$method`, llama reflexivamente (a través del método `getMethod` de `reflectionClass`), al método `bind_param` de la clase `mysqli`. En la siguiente instrucción, a través del método `invokeArgs` de `ReflectionClass`, le pasa por referencia a `bind param`, los datos a ser enlazados con la sentencia preparada (almacenados en el array `$data`). Podría decirse que `invokeArgs`, se comporta de forma similar a `call_user_func_array()`.

Enlazar resultados de una consulta de selección:

```
protected static function get_data($fields) {
    $method = self::$reflection->getMethod('bind_result');
    $method->invokeArgs(self::$stmt, $fields);
    while(self::$stmt->fetch()) {
        self::$results[] = unserialize(serialize($fields));
    }
}
```

Este método es uno de los más “complejos y rebuscados”, que incluso cuenta con algunas “pseudo-magias” un tanto “raras” como el uso de las funciones `serialize` y `unserialize` en la la misma instrucción. Pero analicémoslo detenidamente.

El parámetro `$fields` será recibido a través del único método público que crearemos en nuestra capa de abstracción (este método, a la vez, recibirá este dato, también como parámetro, pero opcional).

Este parámetro, será un array asociativo, donde las claves, serán asociadas al nombre de los campos, y el valor de esas claves, al dato contenido en el campo.

Si tuviese la siguiente consulta SQL:

```
SELECT nombre, descripcion, precio FROM producto WHERE categoria = ?
```

Mi array asociativo, podría paracerse al siguiente:

```
$fields = array("Producto" => "",
                "Descripción" => "",
                "Precio" => "");
```

`mysqli->bind_result()` enlazará el campo producto.nombre a la clave Producto,

producto.descripcion a la clave Desripción y producto.precio a la clave Precio.

Las instrucciones:

```
$method = self::$reflection->getMethod('bind_result');
$method->invokeArgs(self::$stmt, $fields);
```

se comportan de forma similar, a sus homónimas en el método set_params. Llama reflexivamente al método `bind_result` de la clase `mysqli` y le pasa por referencia, el array `$fields`.

En el bucle `while`, estamos asociando iterativamente los datos obtenidos a nuestra propiedad pública `$results`. Pero ¿cómo lo hacemos? ¿para qué serializar y deserializar los datos?:

```
while(self::$stmt->fetch()) {
    self::$results[] = unserialize(serialize($fields));
}
```

En cada iteración, `stmt->fetch` nos está retornando nuestro array `$fields`, asociado al registro de la tabla, sobre el cuál se está iterando. Es decir, que en cada iteración, `stmt->fetch` nos retornará algo como esto:

```
// contenido del array $fields
array("Producto" => "HD Magazine",
      "Descripción" => "Magazine digital de edición mensual sobre Software
                        Libre, Hacking y Programación",
      "Precio" => "0.00");
```

Pero nuestro array `$fields`, ha sido pasado por referencia. Ergo, en cada iteración, su contenido será modificado.

Si a mi propiedad estática `$results`, le agrego como elemento, un array que está siendo modificado por referencia en el momento que lo estoy asignando a mi propiedad estática, mi propiedad estática, será también, modificada en cada iteración.

Para prevenir eso, serializo mi array `$fields` y lo almaceno en `$results` serializado. Pero como luego necesitará recuperarlo, ahorro un paso y lo deserializo en la misma iteración. Al serializarlo, estoy "mágicamente" emulando una "inmutabilidad" de los datos asociados.

Cerrar conexiones abiertas:

```
protected static function finalizar() {
    self::$stmt->close();
    self::$conn->close();
}
```

Un método público que ejecute todas las acciones:

```
public static function ejecutar($sql, $data, $fields=False) {
    self::$sql = $sql;  # setear la propiedad $sql
    self::$data = $data;  # setear la propiedad $data
    self::conectar();  # conectar a la base de datos
    self::preparar();  # preparar la consulta SQL
    self::set_params();  # enlazar los datos
    self::$stmt->execute();  # ejecutar la consulta
    if($fields) {
        self::get_data($fields);
    } else {
        if(strpos(self::$sql, strtoupper('INSERT')) === 0) {
            return self::$stmt->insert_id;
        }
    }
    self::finalizar();  # cerrar conexiones abiertas
}
```

La estructura de control de flujo condicional, que utiliza el método `ejecutar()`, es la encargada de discernir si se trata de una consulta de "lectura" a la base de datos para así, llamar al método `get_data`, o si se trata de una consulta de "escritura" (INSERT, UPDATE o DELETE). En ese caso, verifica si es una escritura de tipo "INSERT" para retornar la id del nuevo registro creado.

Código completo de la capa de abstracción

```
class DBConnector {

    protected static $conn;
    protected static $stmt;
    protected static $reflection;
    protected static $sql;
    protected static $data;
    public static $results;

    protected static function conectar() {
        self::$conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
    }

    protected static function preparar() {
        self::$stmt = self::$conn->prepare(self::$sql);
        self::$reflection = new ReflectionClass('mysqli_stmt');
    }

    protected static function set_params() {
        $method = self::$reflection->getMethod('bind_param');
        $method->invokeArgs(self::$stmt, self::$data);
    }

    protected static function get_data($fields) {
        $method = self::$reflection->getMethod('bind_result');
        $method->invokeArgs(self::$stmt, $fields);
```

```
            while(self::$stmt->fetch()) {
                self::$results[] = unserialize(serialize($fields));
            }
        }

        protected static function finalizar() {
            self::$stmt->close();
            self::$conn->close();
        }

        public static function ejecutar($sql, $data, $fields=False) {
            self::$sql = $sql;
            self::$data = $data;
            self::conectar();
            self::preparar();
            self::set_params();
            self::$stmt->execute();
            if($fields) {
                self::get_data($fields);
            } else {
                if(strpos(self::$sql, strtoupper('INSERT')) === 0) {
                    return self::$stmt->insert_id;
                }
            }
            self::finalizar();
        }

    }
```

¿Cómo utilizar la capa de abstracción creada?

En todos los casos, siempre será necesario invocar estáticamente al método ejecutar, pasándole al menos dos parámetros: la sentencia SQL a preparar y un array con los datos a enlazar a la sentencia SQL preparada:

```
$sql = "INSERT INTO productos
        (categoria, nombre, descripcion, precio)
        VALUES (?, ?, ?, ?)";

$data = array("isbd",
              "{$categoria}", "{$nombre}", "{$descripcion}", "{$precio}");

$insert_id = DBConnector::ejecutar($sql, $data);
```

Cuando se tratare de una consulta de selección, se deberá adicionar un tercer parámetro, el cuál será un array asociativo, cuyas claves, serán asociadas a los campos de la tabla:

```
$sql = "SELECT nombre, descripcion, precio
        FROM   productos
        WHERE  categoria = ?";

$data = array("i", "{$categoria}");
$fields = array("Producto" => "", "Descripción" => "", "Precio" => "");
DBConnector::ejecutar($sql, $data, $fields);
```

```
print_r(DBConnector::$results);
```

La última línea, muestra con un `print_r()` como acceder a los resultados de la consulta, mediante la propiedad estática `$results`.

¿Por qué Python?

PYTHON

Mucho(a)s programadore(a)s se preguntan por qué preferir Python a otros lenguajes de programación, cuáles son sus ventajas y los muchos 'porqués' que lo hacen un lenguaje genial, además conoceremos más de esta filosofía con el Zen de Python.

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**

Python, no, no la serpiente, es el lenguaje de programación (nombre que proviene de la afición de su creador por el grupo de humoristas británicos Monty Python) que ya hace un buen tiempo viene demostrando su magnificidad al momento de desarrollar. En estos tiempos existe una gran cantidad de lenguajes de programación, muchos de ellos simples, complicados, fáciles, enredados y otros más; pero existen también muchos porqués hace ya un muy buen tiempo la gente prefiere aprender python y sobretodo usarlo para grandes proyectos de desarrollo.

Python tiene una sintaxis simple, clara y sencilla, el tipado es dinámico, posee una gran cantidad de librerías disponibles y sobretodo cuenta con una gran potencia. Podemos

mencionar casos de éxito con Python, por ejemplo: Google, Yahoo, la NASA, YouTube, entre otros más.

Veamos una comparación simple entre lenguajes (Python y C++), por ejemplo, si queremos crear un programa para calcular e imprimir la suma de 1+2+3+4+5+...+50:

Python	C++

`h = range(1, 51)`
`print sum(h)`
	`void main ()`
	`{`
	`  int suma;`
	`  for (int i=0, i<=50, i++)`
	`  suma=suma+i;`
	`  cout<<suma;`
	`};`

Como podemos notar la diferencia en la cantidad de líneas de código es muy notable y pues este ya es un punto muy favorable para Python y que pueda ser llamativo a los demás.

Algunas de las características notables de Python son: [0]

- Usa una sintaxis elegante, haciendo de los programas que escribe más fáciles de leer.
- Es un lenguaje fácil de usar que hace simple que su programa trabaje. Esto hace a Python ideal para el desarrollo de prototipos y otras tareas de programación ad-hoc, sin comprometer la mantenibilidad.
- Viene con una gran biblioteca estándar que soporta muchas tareas de programación comunes como la conexión a servidores web, búsqueda de texto con expresiones regulares, leer y modificar archivos.
- El modo interactivo de Python hace que sea fácil de probar fragmentos cortos de código. También hay un entorno de desarrollo incluido llamado IDLE.
- Se puede extender fácilmente añadiendo nuevos módulos implementados en un lenguaje compilado como C o C++.
- También puede ser embebido en una aplicación para proporcionar una interfaz programable.
- Se ejecuta en muchas computadoras y sistemas operativos diferentes: GNU/Linux, Windows, MacOS, muchas marcas de Unix, OS/2, ...
- Es software libre en dos sentidos. No cuesta nada descargar o usar Python, o incluirlo en su aplicación. Python también puede ser libremente modificado y redistribuido, ya que mientras el lenguaje tiene derechos de autor está disponible bajo una licencia de código abierto.

Algunas características del lenguaje de programación de Python son:

- Una variedad de tipos de datos básicos están disponibles: números (coma flotante, complejo y enteros largos de longitud ilimitada), cadenas (ASCII y Unicode), listas y diccionarios.
- Python soporta programación orientada a objetos con clases y herencia múltiple.
- El código puede ser agrupados en módulos y paquetes.
- El lenguaje soporta excepciones de crecimiento y captura, lo que resulta en el manejo de errores más limpio.
- Los tipos de datos tienen tipado fuerte y dinámico. Mezcla tipos incompatibles (por ejemplo, tratar de añadir una cadena y un número) produce una excepción elevada, por lo que los errores son capturados pronto.
- Python tiene funciones de programación avanzadas como generadores y listas de comprensión.
- La gestión de memoria automática de Python lo libera de tener que asignar manualmente y liberar memoria en su código.

> *La alegría de codear en Python debería estar en verlo corto, conciso, con clases legibles que expresen mucha acción en una cantidad pequeña de código claro – no en páginas y páginas de código trivial que aburre al lector hasta la muerte.*
> *-- Guido Van Rossum (creador de Python)*

Zen de Python

El Zen de Python fue escrito por Tim Peters, el cuál cuenta con 20 aforismos[1]:

1. Hermoso es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.

6. Disperso es mejor que denso.
7. La legibilidad cuenta.
8. Los casos especiales no son suficientemente especiales como para romper las reglas.
9. Aunque lo pragmático gana a la pureza.
10. Los errores nunca deberían dejarse pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Cuando te enfrentes a la ambigüedad, rechaza la tentación de adivinar.
13. Debería haber una — y preferiblemente sólo una — manera obvia de hacerlo.
14. Aunque pueda que esa manera no sea obvia a primera vista a menos que seas holandés. (Nota: Guido van Rossum es holandés)
15. Ahora es mejor que nunca.
16. Aunque muchas veces nunca es mejor que *ahora mismo*.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es sencilla de explicar, puede que sea una buena idea.
19. Los espacios de nombres son una gran idea — ¡tengamos más de esas!

Y te preguntarás: ¿Dónde está el número 20?, pues hay muchas teorías para explicar el hecho de que no aparezca en la lista (explícitamente hablando), no hay una verdad absoluta, pero la más aceptable es el significant whitespace, que los espacios son importantes y pues justamente despues de los 19 hay un espacio en blanco que no es imprimible (pero si está impreso). Si usas python 2.1.2 en adelante puedes leer la filosofía en cualquier lugar en el que estés, tan solo con poner la siguiente línea de código (esto es considerado como un huevo de pascua de regalo al encontrarlo de manera tan fácil con un simple comando).

```
>>> import this
```

Y obtendrás esto:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
```

```
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Anímate a comprobar por ti mismo cuán genial es Python y también a poner en práctica el Zen de Python e inclusive tenerlo como filosofía para cualquier proyecto de desarrollo que emprendas.

Referencias

[0] Python Programming Language – Official Website [en línea]: <http://wiki.python.org/moin/BeginnersGuide/Overview> [Consulta: 06 de Octubre de 2012]

[1] Python Programming Language – Official Website [en línea]: <http://www.python.org/dev/peps/pep-0020/> [Consulta: 06 de Octubre de 2012]

Empezando con Google App Engine

PYTHON

Google App Engine (GAE) es una plataforma como servicio que nos permite alojar aplicaciones web y correrlas sobre la infraestructura de Google. En este artículo veremos algunas de las ventajas que nos proporciona y un pequeño tutorial para conocer sus funciones básicas.

Escrito por: **Filly** (Programadora)

Estudiante de programación, con habilidad para el diseño. Activista de la cultura y el software libres. Interesada en el desarrollo web y de videojuegos. **Miembro de Python Argentina** y **GrULiC** (Grupo de Usuarios de Software Libre de Córdoba).

Webs:
Blog: http://www.missfilly.com.ar

Redes sociales:
Twitter: **@MissFillys**
Identi.ca: **@MissFilly**

Supongamos que queremos desarrollar una aplicación web por diversión y ponerla a disposición de otros usuarios, pero no pagar por su alojamiento. O si lo hacemos con fines comerciales y no estamos seguros de qué tan exitosa puede llegar a ser, y por lo tanto no sabemos si es redituable pagar por un servidor. ¿Qué tan genial sería poder alojarla de forma gratuita, fácilmente, y probarla con usuarios reales? Con Google App Engine, de forma gratuita contamos con escalabilidad automática y una cuota de 1 GB de almacenamiento de código y datos estáticos para nuestra aplicación, disponemos de un entorno de desarrollo local para testearla y de una base de datos (Google App Engine Datastore) y tenemos la posibilidad de utilizar un dominio personalizado, entre otras muchas opciones. Si nuestra aplicación es exitosa o queremos ampliarla, y sobrepasamos la cuota[3] de GAE (cantidad de recursos que se nos permite consumir por día), podemos habilitar su facturación, que tiene precios bastante acotados, para adquirir más recursos.

GAE utiliza Python o Java, y actualmente tiene soporte experimental para Go. También incluye librerías de terceros, por ejemplo (en el caso de proyectos en Python 2.7) Django,

3 http://developers.google.com/appengine/docs/quotas

Jinja2, lxml y PyCrypto, entre otras, aunque a la hora de utilizarlas debemos tener muy en cuenta cuáles son sus versiones soportadas y las posibles modificaciones que puedan haber tenido al ser adaptadas a GAE.

Aprender a desarrollar para esta plataforma es relativamente sencillo, y una de las grandes ventajas (que encuentro personalmente) es la excelente documentación con la que cuenta.

Corriendo nuestro 'Hello, world!'

Lo primero que necesitamos hacer para empezar a desarrollar una aplicación para GAE es descargar el SDK[4], en este caso el SDK para Python y Linux. Una vez extraído, encontramos dentro una carpeta llamada `new_project_template`, que contiene, como su nombre lo indica, un template con los archivos básicos de los que constará nuestra aplicación. Yo coloqué el SDK en mi `home`, copié el template en el mismo directorio y lo renombré "mi_primer_aplicación":

```
filly@elric ~ $ cp -r google_appengine/new_project_template/ /home/filly/
filly@elric ~ $ mv new_project_template/ mi_primer_aplicacion/
```

Probar nuestro `Hello, world!` (que ya viene en el template básico) **es tan fácil como** ejecutar el servidor web de desarrollo:

```
filly@elric ~ $ python google_appengine/dev_appserver.py
~/new_project_template/
```

Y listo, ya tenemos nuestra aplicación corriendo en un servidor local. Podemos verla ingresando a `localhost:8080` desde nuestro browser.

¿Qué pasó?

Si abrimos el archivo de Python `main.py`, que contiene el `Hello, world!` dentro de nuestra aplicación, veremos el siguiente código:

```
import webapp2

class MainHandler(webapp2.RequestHandler):
    def get(self):
        self.response.out.write('Hello world!')

app = webapp2.WSGIApplication([('/', MainHandler)],
                              debug=True)
```

Vemos que contamos con la URL '/', que corresponde a la clase `MainHandler`, la cual hereda de `webapp2.RequestHandler`, un handler de requests genérico de Google. El método `get` usa `self.response`, el objeto de respuesta global utilizado por este framework.

4 http://developers.google.com/appengine/downloads

Si quisiéramos crear otra URL para nuestra aplicación, podríamos reemplazar el código por:

```
import webapp2

class MainHandler(webapp2.RequestHandler):
    def get(self):
        self.response.out.write('Hello world!')

class OtherPageHandler(webapp2.RequestHandler):
    def get(self):
        self.response.out.write('This is a different page.')

app = webapp2.WSGIApplication([('/', MainHandler),
                               ('/other', OtherPageHandler)],
                              debug=True)
```

Ahora podemos acceder a la nueva página en `localhost:8080/other` desde un browser.

Templates

Para mantener nuestro código limpio, contamos con el sistema de templates de GAE, que nos permite tener el HTML separado del código de la aplicación propiamente dicha, utilizando una "sintaxis especial para indicar dónde aparecen los datos de la aplicación"[5].

Para probarlo, primero creamos una carpeta `templates` en el directorio de nuestro programa, y en su interior un archivo `index.html` con el siguiente código:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mi primer aplicación</title>
  </head>
  <body>
    <h1>¡Hola, mundo!</h1>
    <p>Mi nombre es {{ name }}.</p>
  </body>
</html>
```

Después, cambiamos el código del archivo `main.py` por:

```
import webapp2
import os

from google.appengine.ext.webapp import template

class MainHandler(webapp2.RequestHandler):
    def get(self):
        path = os.path.join(os.path.dirname(__file__),
'templates/index.html')
        template_values = {'name': 'Filly'}
        self.response.out.write(template.render(path, template_values))
```

5 https://developers.google.com/appengine/docs/python/gettingstarted/templates

```
app = webapp2.WSGIApplication([('/', MainHandler)],
                              debug=True))
```

El módulo `webapp2` contiene el motor de templates de Django[6], y por lo tanto utiliza la misma sintaxis. La función `template.render(path, template_values)` toma la ruta del archivo del template y un diccionario, cuyas keys son los nombres de las variables a ser reemplazadas en el template (indicados entre llaves dobles en el mismo, como en el caso de `{{ name }}`) con sus respectivos valores.

La Datastore

Google App Engine Datastore nos "ofrece un almacenamiento sólido y escalable para las aplicaciones web"[7]. Los datos de una aplicación se escriben en entidades, cada una de las cuales cuenta con una id que la identifica (ya sea asignada automáticamente por la Datastore o creada por nosotros).

En GAE, todas las consultas deben ser indexadas previamente, y se utiliza una versión simplificada de SQL, denominada GQL.

Un formulario básico

Vamos a crear una página con un formulario básico para nuestra aplicación, que nos permita guardar en la Datastore los datos ingresados, y además nos muestre los registros existentes.

Para empezar, en la carpeta `templates` creamos otro archivo, llamado, por ejemplo, `form.html`, con este código:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mi primer aplicación</title>
  </head>
  <body>
    <h1>Formulario</h1>
    <h2>Ingresar datos:</h2>
    <form method="post">
      <p>
        <label for="firstname">Nombre: </label>
        <input type="text" name="firstname">
      </p>
      <p>
        <label for="lastname">Apellido: </label>
        <input type="text" name="lastname">
      </p>
      <div class="error">{{ error }}</div>
      <input type="submit" value="Enviar">
    </form>
```

6 https://docs.djangoproject.com/en/1.4/ref/templates/api/
7 https://developers.google.com/appengine/docs/python/datastore/overview

```
    <h2>Datos existentes:</h2>
    <div>
      {% for person in people %}
      <p>{{ person.last_name }}, {{ person.first_name }}</p>
      {% endfor %}
    </div>
  </body>
</html>
```

Los statements encerrados entre '{%' y '%}' son propios de la sintaxis del sistema de templates de Django.

A continuación, necesitamos crear el handler para nuestro formulario, incluido su método `post` (para cuando los datos del formularios son enviados):

```
class FormHandler(webapp2.RequestHandler):
    def get(self, first_name='', last_name='', error=''):
        people = db.GqlQuery('SELECT * FROM Person ORDER BY last_name')
        path = os.path.join(os.path.dirname(__file__),
                            'templates/form.html')
        template_values = {'first_name': first_name,
                           'last_name': last_name, 'error': error,
                           'people': people}
        self.response.out.write(template.render(path, template_values))
    def post(self):
        first_name = self.request.get('firstname')
        last_name = self.request.get('lastname')
        if first_name and last_name:
            p = Person(first_name = first_name, last_name = last_name)
            p.put()
            self.get(first_name = first_name, last_name = last_name)
        else:
            self.get(error = 'No completaste alguno de los campos.')
```

Para crear una entidad en nuestra base de datos, utilizamos una clase que hereda de `db.Model`, que importaremos al principio de nuestra aplicación. Entonces, nuestro archivo `main.py` se verá de esta forma:

```
import webapp2
import os

from google.appengine.ext.webapp import template
from google.appengine.ext import db

class MainHandler(webapp2.RequestHandler):
    def get(self):
        self.response.out.write('Hello world!')


class FormHandler(webapp2.RequestHandler):
    def get(self, first_name='', last_name='', error=''):
        people = db.GqlQuery('SELECT * FROM Person ORDER BY last_name')
        path = os.path.join(os.path.dirname(__file__),
                            'templates/form.html')
        template_values = {'first_name': first_name,
                           'last_name': last_name,
```

```
                            'error': error, 'people': people}
        self.response.out.write(template.render(path, template_values))

    def post(self):
        first_name = self.request.get('firstname')
        last_name = self.request.get('lastname')
        if first_name and last_name:
            p = Person(first_name = first_name, last_name = last_name)
            p.put()
            self.get(first_name = first_name, last_name = last_name)
        else:
            self.get(error = 'No completaste alguno de los campos.')


class Person(db.Model):
    first_name = db.StringProperty(required = True)
    last_name = db.StringProperty(required = True)


app = webapp2.WSGIApplication([('/', MainHandler),
                               ('/form', FormHandler)],
                              debug=True)
```

Ahora, cuando nos dirigimos a `localhost:8080/form` en nuestro browser, completamos los campos del formulario y presionamos el botón “Enviar”, vemos nuevamente el formulario, con los datos que acabamos de ingresar debajo de “Datos existentes:”. Si agregamos más registros, éstos se listarán junto con los anteriores, respetando el orden alfabético del campo “apellido”.

¿Qué pasó?

El código que acabamos de escribir funciona de la siguiente manera:

1. Cuando presionamos el botón enviar se ejecuta nuestro método `post`, que toma los valores ingresados en el formulario y los guarda en sus respectivas variables (`first_name` y `last_name`, es decir, nombre y apellido).
2. Como necesitamos tanto un nombre como un apellido para la persona que guardaremos en nuestra Datastore (`required = True`, en la clase `Person`, hace que sea requerido), el método `post` chequea que ninguna de las variables sea un string vacío, y luego crea un nuevo registro en la entidad `Person`.
3. Si falta completar alguno de los campos (una de las variables es un string vacío), se renderiza nuevamente el formulario, esta vez pasando el correspondiente error a través de `template_values['error']`, cuyo valor es usado para reemplazar `{{ error }}` en el template.
4. Cada vez que se renderiza el formulario, el método `get` repasa todos los registros existentes, y por cada uno de ellos el statement `{% for person in people %}` del template escribe un párrafo de HTML con los datos de la persona.

Conclusión

Una vez hemos testeado nuestra aplicación y queremos subirla, todo lo que resta es seguir los pasos indicados en la sección correspondiente[8] de la documentación de GAE.

Ahora que hemos visto un pantallazo muy general de esta plataforma, lo que nos queda por hacer es pensar en lo que queremos para nuestra aplicación, ¡y sentarnos a codear!

The Hitchhiker Pythonits's Guide to the Galaxy

PYTHON

En este artículo haremos el descabellado intento de plotear una galaxia espiral morfológicamente "pseudo correcta", para esto haremos uso/abuso de Python seguido de sus secuaces Scipy y Mayavi. Utilizando hasta el cansancio funciones lambda, mapeos y filtros entre otros yuyos.

Escrito por: **Celia Cintas** (Licenciada en Informática)

Licenciada en Informática (UNPSJB), actualmente realizando **Doctorado en Procesamiento de Imágenes** (UNS), Docente (UNPSJB), Intento de sysadmin (CC) y code monkey el resto de las horas :). Pythonera por defecto con alarmantes inclinaciones hacia Prolog y otras herejías.

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

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

8 https://developers.google.com/appengine/docs/python/gettingstarted/uploading

Se debe aclarar que el titulo reside en ingles porque no existe traductor humano que pueda llevar a otro idioma este título sin mutilar su sentido humorístico. Antes de meternos completamente en el código, debemos pensar como esto es posible, para desenmarañar esto debemos tener dos conceptos base muy en claro; Qué es un *Autómata Celular*? Y recordar que eran las *Coordenadas Polares*. Se promete que en un tiempo *x* el lector *y* podrá comprender para que son necesarios estas definiciones.

Entre Autómatas y Coordenadas

Los Autómatas celulares son ampliamente utilizados en el ámbito científico, en procesamiento de imágenes (y seguramente muchos casos más), ya que con modelos simplistas se puede llegar a simular/trabajar comportamientos sumamente complejos. Lo cual a la hora de programar nos otorga una placidez invaluable.

> *Los Autómatas Celulares proveen un modelo matemático, simple, discreto y determinista para sistemas biológicos, físicos computacionales y más. A pesar de su simple construcción, los Autómatas Celulares han demostrado ser capaces de reproducir/generar comportamientos complejos.*
> *Wolfram 1982*

Para darnos una idea gráfica, pensemos en una grilla (2D), en el cual cada celda puede tener un estado dado en un tiempo discreto t y este estado depende, visto de la forma más simple, del estado de sus "vecinos", es decir, de las celdas que lo rodean.

La teoría de autómatas celulares excede los horizontes de este articulo pero en [1] se pueden ver gran cantidad de *papers* de su principal propulsor *Stephen Wolfram*.

Lo que respecta a coordenadas polares, solo debemos tener en cuenta las ecuaciones que nos permitan pasar las mismas a coordenadas cartesianas.

$$x = r * \cos(\theta) \qquad y = r * \sin(\theta)$$

Donde *r* es el radio y θ su ángulo.

Ahora porque usar coordenadas polares si nuestros autómatas son una grilla cuadriculada ?, la respuesta es simple, nuestro autómata tiene una vecindad circular, si uno lo piensa tiene bastante sentido ya que la morfología que estamos buscando es circular.

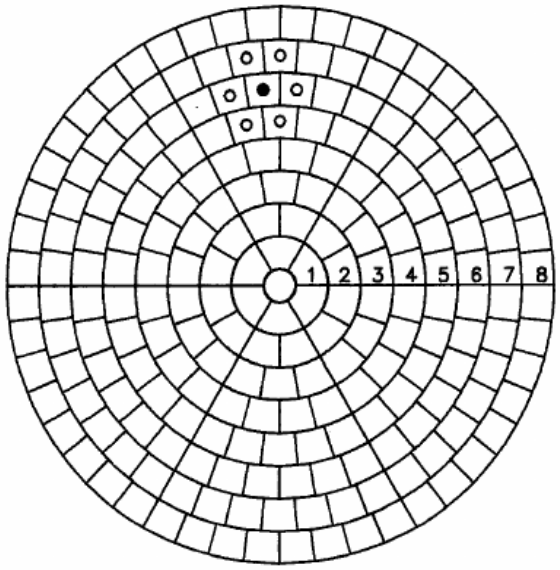
Autómatas y Galaxias

Un autómata celular en 2D puede ser usado como un eficiente constructor de galaxias y una herramienta para observar su evolución. Este modelo mantiene la estructura espiral usando como entrada datos sobre formación estelar, posición, tamaño y velocidad y la rotación de la galaxia. [2]

A continuación daremos los pasos para la construcción de nuestra galaxia tomando como base los dados en [2].

1. Las celdas pueden tener 15 posibles estados (vida de una formación estelar/dt), con cero denotamos la ausencia de estrellas o formaciones.
2. Dada una celda activa, se examina su vecindad y estas tienen una probabilidad de iniciarse de 16 a 22%, en este caso particularmente se trabajo con 18%.
3. Cada celda tendrá tres datos cruciales, radio y ángulo, para detectar su posición y un estado que nos indica en que etapa se encuentra nuestra formación.
4. Luego de examinarlas incrementamos en uno a todas las estrellas activas para denotar su edad que se verá reflejada en una paleta de colores, y aplicamos una función de rotación.

En la sección anterior hablábamos de coordenadas polares, en este punto podemos vislumbrar cual será su utilidad. Para poder evaluar cada celda debemos conocer su posición y ya que hablamos de generar galaxias del tipo espiral estas coordenadas demuestran ser las indicadas.



En la Figura tomada de [3] podemos observar un ejemplo de la vecindad a la que nos referimos.

El modelo tomado de [3] que consta de N círculos concéntricos, cada anillo es divido en $6N$ regiones.

Cada celda tiene normalmente 6 vecinos, si la celda se encuentra activa, es decir, si se generó una formación de estrellas, a medida que avance la simulación también lo harán sus estado (tiempo de vida o mejor dicho edad).

La obtención de los patrones no es del todo trivial, ya que estas no rotan en forma rígida.

La velocidad de rotación varía rápidamente cerca del centro de la galaxia y en radios

superiores se vuelve bastante plana. En toda la zona perteneciente a los brazos de la galaxia la velocidad circular es independiente del radio, lo que nos indica que la velocidad circular es independiente del radio y que la velocidad angular no es constante y decrese en forma de $\frac{1}{r}$.[3]

Implementación: *Galaxpy*

Primero daremos un breve *paneo* de las herramientas utilizadas:

- **Python** es un lenguaje de programación que permite trabajar con mayor rapidez e integrar sistemas con mayor eficacia. Se puede aprender a usar Python y obtener beneficios casi inmediatos en la productividad y reducir los costos de mantenimiento. Para más información http://python.org/
- **Prymatex** es un editor basado en PyQt, que soporta Bundles de TextMate, contiene Snippets, resaltado de sintaxis, entre otras bondades. Para más información http://prymatex.org/
- **Ipython** es una shell interactiva que añade funcionalidades extra al modo interactivo incluido con Python, tales como resaltado de líneas y errores mediante colores, una sintaxis adicional para el shell, autocompletado mediante tabulador de variables, módulos y atributos; entre otras. Es un componente del paquete SciPy. Para más información http://ipython.org/
- **NumPy** permite manipular de manera rápida y eficiente arreglos numéricos tales como vectores y matrices (pero también arreglos multidimensionales de rango arbitrario). Para más información http://numpy.scipy.org/
- **SciPy** contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería. Está dirigida al mismo tipo de usuarios que los de aplicaciones como MATLAB, GNU Octave, y Scilab. Para más información http://www.scipy.org/
- **Mayavi2** es un programa interactivo (en forma opcional) que permite elaborar gráficos en 3D de los datos científicos en Python con scipy. Es el sucesor de MayaVi para la visualización 3D. Para más información http://mayavi.sourceforge.net/

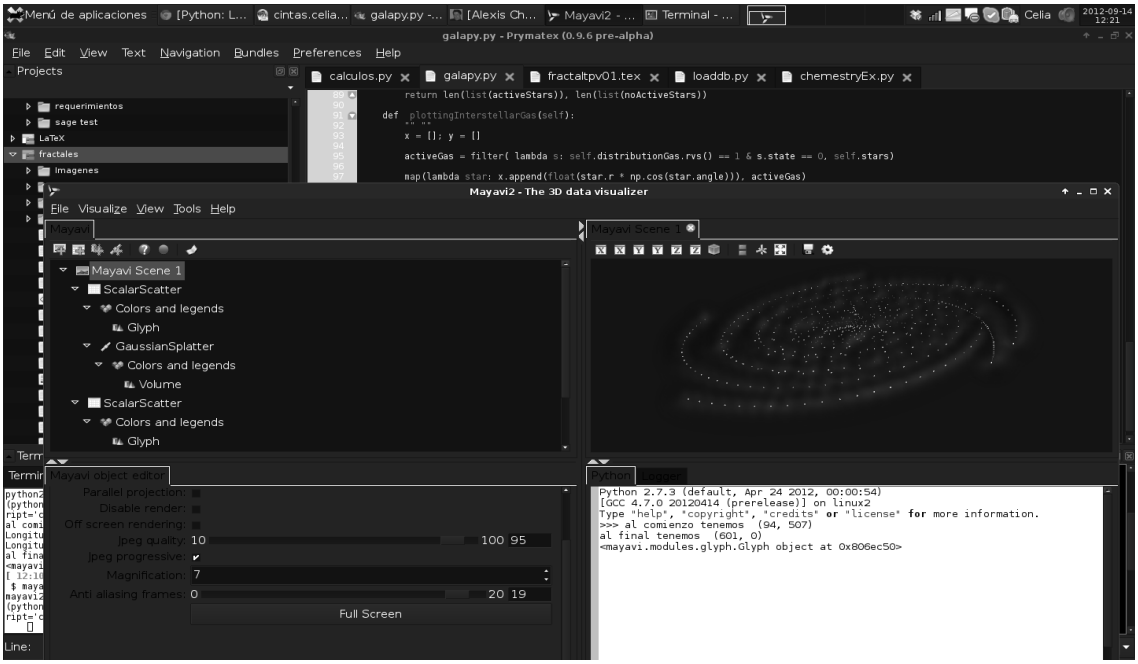
Ahora vayamos al código, este puede bajarse de [4] en este artículo solo veremos fragmentos cuasi interesantes.

Para la distribución de las estrellas tomamos la función detallada en [2]. En el código se puede ver flejada en:

```
....
import scipy.stats.distributions as dis
....
```

```
class Galaxy:

  def __init__(self, radio, neighborhood):
    """Initialize the galaxy with some radius
    and the chosen neighborhood """
    self.distributionFuncStars = dis.rv_discrete(name='custm',
                                    values=[(0, 1),(0.84, 0.16)])
    self.distributionGas = dis.rv_discrete(name='custm_gas',
                                    values=[(0,1),(0.3,0.7)])
....
....
```

Para agregar un poco más de aleatoriedad se agrego la funcion de distribución *distributionGas*. En un comienzo se trató de tomar la función "real" pero se tuvo que modificar para que esta aleatoriedad aparezca. Podemos ver los resultados generados tomando estas funciones de distribución en la figura:



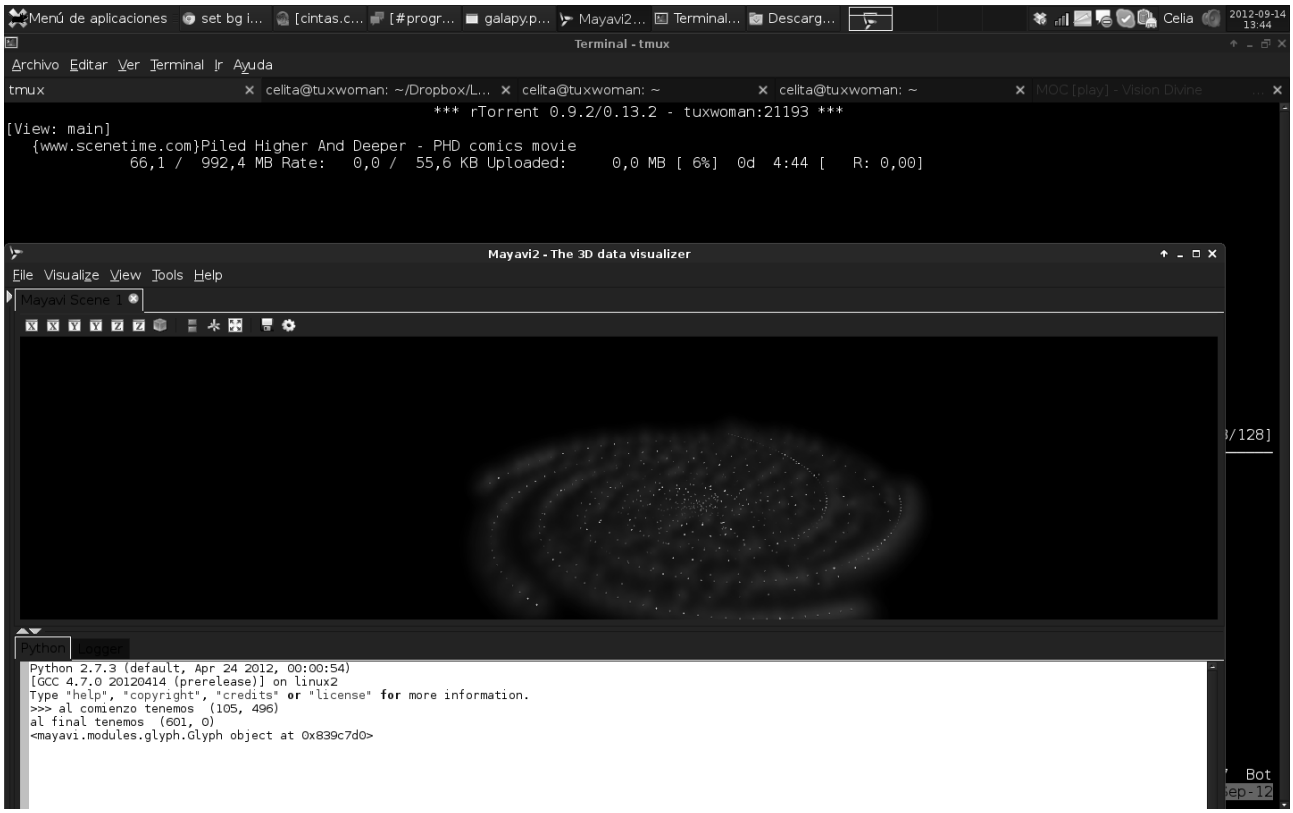
Para reflejar la velocidad angular explicada en la sección anterior se implemento dentro de la clase *Galaxy* la siguiente función.

```
    ....

 def growthFunction(self, star):
    """Increases growth of each star"""
    if star.state > 15:
        star.state = 0
    star.state += 1
    try:
        #set angular velocity
        star.angle = star.angle + 1 / float(star.r)
    ....
    ....
```

```
        ....
```

La forma en que se rastrillo la galaxia, es decir, ver el estado de los vecinos aledaños a un cúmulo estelar, para saber si se debía crear un nuevo cúmulo o no, se muestra a continuación en el código:

```
        ....
        ....

    def scanning(self):
        """See state of stars and change formations"""
        for s in self.stars:
            localNeighborhood = ifilter(lambda star: star[1] == star[0].r +
                                        1,izip(self.stars, repeat(s.r,
                                        len(self.stars))))
            map(self.birthFunction, localNeighborhood)
        #At this line we age the Galaxy.
        map(self.growthFunction, self.stars)
        ....
```



Referencias & Links

[1] Stephen Wolfram: Articles on Cellular Automata

[2] Morphology and Dynamics of Galaxies with Cellular Automata, Dr. Brian R. Kent, National Radio Astronomy Observatory.

[3] Percolation Model of Galactic Structure. P.E. Seiden and L. S. Seiden and L. S.

Schulman. Adv. Phys., 39:1-54, 1990.

[4] Código Fuente

GNU/Linux & Servers: Tricks & Tips

TIPS & TRICKS

Trucos y consejos para resolver problemas frecuentes de forma rápida, sencilla y por qué no, divertida :)

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
Agile Coaching: www.eugeniabahit.com

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

Navegar en Internet saltando restricciones utilizando a Google como proxy

Muchas veces, queremos ingresar a un sitio Web y no podemos. Ya sea porque la red desde la cual estamos intentando ingresar bloquea el acceso a dicho dominio o peor aún, porque nuestra IP se encuentra en un rango de Ips restringidas para acceder a dicho dominio (generalmente, debido a que por órdenes judiciales, el proveedor de Internet, servidor o el mismo titular del dominio, se vio forzado a restringir el acceso, por ejemplo, a Ips de ciertos países).

Este problema, generalmente **se resuelve utilizando un proxy**, pero no siempre tenemos la posibilidad de configurar uno. Sin embargo, **el servicio de traducción de**

sitios Web de Google (e incluso, otros servicios de traducción), de forma indirecta, **actúa como un proxy**, ya que para mostrarnos dicho sitio Web traducido, nosotros accedemos a Google y es Google quien accede al sitio Web "con su IP" y no con la nuestra.

Entonces **¿cómo utilizar a Google como proxy?** Muy simple. En la barra de direcciones del navegador, utilizaremos la siguiente URI para acceder al sitio Web restringido:

```
http://translate.google.com/translate?sl=en&tl=es&u=http://www.restringido.com
```

Dónde `http://www.restringido.com` será la URL del sitio Web a que queremos acceder. Los dos primeros parámetros -NECESARIOS- de la URI anterior, `sl` y `tl` representan el idioma de origen y destino respectivamente. Para leer el sitio Web en su idioma original (sin traducción), al primer parámetro (idioma de origen, sl) se le puede indicar un idioma que NO sea el verdadero. Por ejemplo, si el sitio Web está en español, a `sl` le indicamos en.

Tener en cuenta que idioma de origen y destino deben diferir. Por eso, si "mentimos" en el idioma de origen, en el de destino, sí podremos indicar el verdadero:

```
http://translate.google.com/translate?sl=es&tl=en&u=http://www.facebook.com
```

Abreviando comandos en el .bashrc

Si ya lograste cansarte de tener que escribir una y otra vez, largas listas de comandos para realizar una acción repetida, te muestro dos opciones para que -desde ahora- puedas evitarlo fácilmente: **alias** y **funciones**.

En el `.bashrc` tienes la posibilidad de crear alias de comandos y funciones. Dicho archivo se encuentra en `/home/tu-usuario/.bashrc`

Alias

Los **alias** pueden servirnos para crear "abreviaciones" de comandos y argumentos que utilicemos con frecuencia. Por ejemplo (una sencilla para que se entienda):

En vez de tener que escribir `ls -lha`, solo escribir `ls`:

```
alias ls='ls -lha'
```

Simple ¿cierto? La sintaxis siempre debe ser:

```
alias abreviacion='lista de comandos y argumentos'
```

Funciones

Para procesos más complejos, podemos utilizar **funciones**, sobre todo, si algunos parámetros no suelen ser siempre iguales:

```
function md5 {
    php -r "echo md5('$1') . chr(10);"
}
```

La función anterior, nos permitirá ejecutar el comando md5 seguido de una cadena a *hashear* y nos imprimirá en pantalla el hash MD5 de dicha cadena, utilizando PHP-CLI. `$1` representa el primer argumento pasado al comando md5.

Jugar con usuarios molestos es más divertido que bloquearlos

Estoy completamente segura de que alguna vez, un usuario molesto de tu Sitio Web, te generó dolores de cabeza y terminaste bloqueando su acceso. Pero ¿para qué amargarte si puedes divertirte?

> *Deja al molesto/a loguearse en tu sistema y diviértete!*

Personalmente, hay dos cosas que me divierten mucho hacer, cuando un usuario "se pone pesado". Una de ellas, es **redirigirlo a algún sitio Web** que nada tenga que ver con el mío, cada vez que detecto que se encuentra logueado en el sistema:

```
if(isset($_SESSION['user'])) {
    if($_SESSION['user'] == 'nombre-del-usuario-molesto') {
        header('Location: http://www.vaticano.va');
        exit();
    }
}
```

Pero otra opción, es **hacerle creer que la página a la que intenta acceder, no existe**:

```
if(isset($_SESSION['user'])) {
    if($_SESSION['user'] == 'nombre-del-usuario-molesto') {
        header('HTTP/1.1 404 Not Found');
        exit();
    }
}
```

Definitivamente, modificando los encabezados HTTP, son muchas las cosas que puedes hacer para divertirte y evitarte dolores de cabezas ;)

Contribuyendo en el equipo de traducción al español de GNOME

SOFTWARE LIBRE

Cómo fue que me animé a contribuir en Malditas Mentiras, en el equipo de traducción al español de GNOME sobre cuán genial es contribuir al equipo y cómo también tu puedes formar parte de esto.

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**

Hace ya ocho meses estoy contribuyendo en Malditas Mentiras, soy parte del equipo de traducción al español de GNOME [0], hasta el momento ya he colaborado en 8 diferentes módulos con traducciones: en la guía de usuario de Pitivi y de Conglomerate, en el contenido estatico de la pagina web de GNOME, en GIMP con una parte de los diálogos y de los menús, en el IU de evolution-kolab y de gnumeric y actualmente estoy traduciendo la sección de funciones de gnumeric[1]).

Uno de mis hobbies (un hobby forever alone :P) es traducir, desde pequeña siempre me agradó la idea de aprender otros idiomas y a medida que aprendía inglés me interesó mucho el tema de las traducciones y se volvió de esta manera en uno de mis pasatiempos favoritos, al principio solo lo hacía en mi entorno más cercano, quizás por alguna necesidad en las clases o por si algún familiar o amigo necesitaba leer algo escrito originalmente en inglés, en español; luego cuando empecé a entrar en el mundo del software libre y del código abierto, también me empezó a interesar la ingeniería de software y ví que podía hacer algo con ese pasatiempo en beneficio de alguna comunidad o proyecto real.

Es así como empiezo a usar GNOME y luego me entero de la existencia y el trabajo en Damned Lies (Malditas Mentiras), esta es la aplicación web usada para gestionar la localización del proyecto GNOME (l10n, este es un numerónimo, una palabra que contiene números, se considera la primera y la última letra de la palabra y las demás letras se reemplazan por el número que representan las omitidas).

Malditas Mentiras[2] está basada en el Framework web Django , ofrece muchas estadísticas (actualizado a cada commit) , ofrece un flujo de trabajo de revisión con diferentes roles (traductor, revisor, committer y administrador), maneja muchos VCS/DVCS (cvs, svn, git, mercurial, bzr) , está organizado en torno a equipos e idiomas, envía mensajes a las diferentes listas de correo del equipo , ofrece canales RSS de las acciones (Vertimus) para cada idioma y equipo , OpenID y autenticación local , detecta cadenas que se rompen por congelación, también permite examinar las estadísticas en tiempo real de los módulos que se vienen traduciendo y gestiona el flujo del trabajo de las traducciones; es software libre y su código fuente está disponible bajo la licencia GPL de GNU [3].

Se dice que el nombre viene de una conocida frase popularizada por Mark Twain que atribuye al primer ministro británico cuando dijo: “Hay tres clases de mentiras: las mentiras, las malditas mentiras y las estadísticas”, era una frase que describe el poder de persuasión de los números y en particular el uso de las estadísticas para reforzar los argumentos débiles [4].

> *“El lector ideal es un traductor. Es capaz de desmenuzar un texto, retirarle la piel, cortarlo hasta la médula, seguir cada arteria y cada vena y luego poner en pie a un nuevo ser viviente”.*
> *Alberto Manguel*

Mi experiencia siendo parte del equipo de traductores al español de GNOME ha sido genial, ya que aparte de que el traducir es un hobby para mi, el sentir que puedes ayudar a los demás a poder usar software que originalmente está en inglés y ver cómo ahora pueden usarlo en español, es gratificante saber que aunque sea una pizca de tu contribución le será de utilidad a alguien más al poder usar software en su idioma nativo. Anímate a ser parte de este proyecto, solo tienes que darte un tiempo para poder contribuir con GNOME; escribir un mail a la lista de correos, presentarte y decir que deseas colaborar es todo lo que necesitas para empezar; toda la información detallada podrás encontrarla en la página oficial de Malditas Mentiras de GNOME[5].

Referencias & Links

[0] http://l10n.gnome.org/teams/es/

[1] http://l10n.gnome.org/vertimus/gnumeric/master/po-functions/es

[2]Live GNOME – Damned Lies https://live.gnome.org/DamnedLies [Consulta: 08 de Octubre de 2012]

[3] Malditas Mentiras – Acerca de http://l10n.gnome.org/about/ [Consulta: 08 de Octubre de 2012]

[4] http://en.wikipedia.org/wiki/Lies,_damned_lies,_and_statistics

[5] http://l10n.gnome.org/

¿La crisis del software?

IT & SOCIEDAD

El uso masivo de dispositivos tecnológicos y servicios a través de internet, ha generado en todo tipo de entornos, necesidades y retos de cara a los hoy por hoy sedientos consumidores de tecnología. Sin embargo ¿Qué tan conscientes son las personas de que cada vez existen menos profesionales entusiasmados por construir aquellas cosas que dan por sentado en su cotidianidad? La crisis profesional de los ingenieros de software está servida.

Escrito por: **Sorey García** (Especialista en Desarrollo de Software)

Arquitecta de software y docente universitaria. **Fundadora de la comunidad Avanet** y miembro de **IEEE Subsección Medellín**. **Líder de investigación y desarrollo** para movilidad y *retail* en **LineaDataScan**.

Webs:
Blog: http://blog.soreygarcia.me

Redes sociales:
Twitter / Identi.ca: **@soreygarcia**
LinkedIn: http://co.linkedin.com/in/soreygarcia

El cambio generacional es evidente, los entornos cotidianos se han visto impactados en mayor o menor medida por la adopción vertiginosa de la tecnología. Algunas situaciones generadas por esta viralización, producto de la fusión entre *la*

movilidad, las redes sociales y el ofrecimiento de servicios en la nube, se han vuelto problemas cotidianos de algunos escenarios. Tal es el caso de la consumerización de las tecnologías de la información, que se ha convertido para las organizaciones en una batalla contra el uso incorrecto del tiempo y recursos, la fuga de información y la utilización de dispositivos no autorizados en los equipos y redes corporativas, entre otros. Pero lejos de estas situaciones que se hacen evidentes por impactar de forma negativa los intereses de algunos pocos con poder y dinero, *otras crisis sobrevienen de manera más lenta y silenciosa desde hace algunos años.*

La tecnología es popular y la construcción de la misma debería ser popular de facto, o eso hace creer el cine, con personajes que se vuelven héroes o grandes villanos por saber programación de computadoras. Para quienes solo nos toca las noches de trasnocho a causa de clientes que no saben lo que quieren, requisitos y legislaciones qué cambian más que el afectado clima o mantener código no documentado de otro programador, es evidente que el tema es *muy ciencia ficción*, pero tal parece que no es así de comprensible para la mayoría de la gente.

El problema consiste en quiénes y cuántos son los que no lo comprenden. Pues bien, la mayoría de personas que desconocen los detalles del *que hacer de la ingeniería de software*, son muchos de los cuales ejercen en las compañías el rol de usuarios y clientes de los proyectos de software, y que exigen en sus aplicaciones *cosas modernas* como las que ven en la televisión, anuncios publicitarios, documentales o artículos de revista, y que en pro de su propia vida moderna y su adopción de nuevas tecnologías, quieren llevar sus obsoletos métodos y procedimientos empresariales hacia entornos vanguardistas, esto sin contar por supuesto, que además desean llevar este mismo estilo de vida a sus tareas personales, diarias y cotidianas. **Debe ser fácil de hacer ¿no?**

Tal pareciera que sí, todos los días en los magazines se leen historias de jóvenes que se hicieron ricos después de vender a grandes compañías, aplicaciones que construyeron en un par de semanas o noches, y que hoy por hoy son un hit entre los usuarios de tecnología, en esa perspectiva, claro que debe parecer ser fácil de hacer.

La respuesta tiene más de un implicado y más de una historia de trayectoria, a usuarios y nuevos profesionales, lamento informarles que no siempre es así, mejor aún, que casi nunca es así. Usar la misma repetida historia de cómo **Angry Birds** se ha hecho la aplicación más popular o como **Mark Zuckerberg** se hizo millonario y famoso, y argumentar que tenemos la posibilidad de ser uno de ellos, es tanto como invitarlos a comprar un boleto de lotería, o mejor, para tener más oportunidad de ganar, motivarlos a comprase dos.

La realidad es un poco diferente para la mayoría y lo es aún más cuando **no** eres un desarrollador *moderno* que persigue iniciar su propio emprendimiento, si no que te enlistas a trabajar en pequeñas o grandes compañías. Motivaciones para esos (nosotros) los más comunes, no hay muchas, y sin embargo los hoy afectados por la carencia de profesionales de software siguen preguntándose *¿por qué no hay suficientes ingenieros de sistemas e informática?*

Si usted apenas se está enterando, esa es la situación. Alrededor del mundo la demanda de profesionales en sistemas calificados, supera la oferta. Las necesidades de las organizaciones no están cubiertas, y los esfuerzos de muchos se centran en motivar los

profesionales en formación, nuevos profesionales e incluso a cualquiera que se crea con capacidades, en que **ser programador de software además de genial es fácil.** Yo estoy de acuerdo con lo primero, pero a quienes aún no conocen de estos menesteres, lamento informarles nuevamente, que solo les están contando una parte de la historia.

Contrario a este discurso proveniente en ocasiones hasta de entidades gubernamentales, cuando las grandes compañías buscan personal para reclutar, las solicitudes van de lo simple a lo irrisorio. Mientras algunos contratan para cargos que requieren de experiencia y conocimiento, a estudiantes o aprendices, otros solicitan personal con experiencia profesional y conocimiento en metodologías y múltiples tecnologías. En ambos casos la remuneración no compensa el nivel de responsabilidad o el perfil solicitado.

Además, hay que ver como muchos de los que en distintos escenarios promueven el discurso de que cualquiera puede ser un programador, no contratarían a alguien empírico en sus propias empresas *a menos que fuese un genio*, puesto que conocen bien el costo de tener que educarlos en procesos un poco más elaborados como los que requieren las fábricas de software o simplemente la empresas que ya sufrieron por años en carne propia, los problemas de haberse permitido tener héroes construyendo software, en vez de profesionales con mentalidad visionaria y que previeran que en el software hay algo más que hacer que el que las aplicaciones funcionen **ahora**, y ese algo más, es que **después**, otros tendrán que mantenerlo y hacerlo funcionar también.

La ingeniería de software, el **que hacer** de los profesionales de software, no alcanzó a madurar y echar raíces y ya es subestimado. El bien intencionado trabajo académico hoy está siendo desechado ante el llamado a las masas a utilizar tutoriales virtuales y algunas horas de dedicación para formarse en el *arte* de hacer software. Creo que malentienden eso de *"el arte",* puesto que a tal declive no nos referimos los profesionales que afectuosamente lo llamamos de esa manera.

Entre tanto el impacto y efecto de este discurso, no mejora ni ayuda a motivar a los adolescentes en que eso de **"hacer software"** deba ser ejercido desde el ámbito profesional. *¿Con que finalidad lo harían si de otro modo pueden construir tecnologías modernas y divertirse más?* Por otro lado *¿Cuántos tutoriales de internet podría leerse uno en los mismos 4 o 5 años que toma la formación profesional?*

Es así como resulta evidente y alarmante, la preocupante disminución de nuevos estudiantes en las carreras técnicas, al punto de en algunos casos llevar a universidades a eliminar las carreras de su oferta académica, ante la imposibilidad de lograr que estas sean sostenibles y rentables con la escasa afluencia de estudiantes.

El discurso más ruidoso actualmente se centra en que marca es la mejor, como lograr crear una empresa en pocos días que te haga rico mágicamente y cuales posibles creaciones, producto del sentido común, no se encuentran patentadas. Pues bien, en el ámbito social, las peleas están lejos de ser cual es o no la mejor tecnología, e incluso si es libre, gratis o paga. Para quienes somos arquitectos de software nada deja más claro que alguien no tiene idea del tema, cuando da argumentos basados en la pasión y no en las realidades de cada escenario, cliente y proyecto. Sin embargo el ruidoso colectivo y sus temas de moda, tiene lejos su atención de las problemáticas de orden mundial como lo resulta hoy en día la falta de fuerza laboral calificada y con experiencia suficiente para

soportar los procesos de negocio de pequeñas y grandes compañías, lo tendrán en mente cuando sea su empresa o emprendimiento el que sea afectado.

La opción, al menos en las pequeñas compañías es sencilla, *es necesario contratar al que lo sabe hacer*, y si tiene un profesional que sabe hacerlo versus algún no profesional que puede dar el *"mismo aparente resultado*" a la mitad de salario, la elección se hace evidente. Los efectos de hacer software que simplemente funciona, vienen con los años, cuando los negocios crecen, las necesidades cambian, las regulaciones imponen nuevas directrices en los sistemas, y entonces llega el mantenimiento y los controles de cambios. Las escenas clásicas llegan con facilidad a la mente de cualquiera que haya pasado por la misma situación: *¿Dónde está la documentación? ¿Qué clase de programador hizo semejante código?* Y la mejor de las expresiones es: *¡Hay que hacerlo desde cero! ¡Nos tomaría más entenderlo, que volver a hacerlo!*

Hagamos una sencilla comparación: *¿Quién en su sano juicio permitiría que alguien le hiciera una cirugía a corazón abierto tan solo porque vio el procedimiento en un video en internet o por que leyó un tutorial que explicaba facilísimo cómo hacerlo?* Resulta muy obvio sentar una posición frente a esta pregunta *¿Por qué no resulta igual frente al escenario tecnológico?*

> *¿Quién en su sano juicio permitiría que alguien le hiciera una cirugía de corazón abierto tan solo porque vio el procedimiento en un video en la Internet o porque lo leyó en un tutorial que explicaba facilísimo como hacerlo?*

Si a usted la medicina y el hacer software no le parecen escenarios comparables, permítase analizar por un segundo, cuales son los aspectos de su vida en los que el software tiene un rol fundamental. Desde los escenarios más sencillos y cotidianos, como las tan de moda aplicaciones en los dispositivos móviles, hasta los entornos sociales más sensibles como la medicina y la banca, hacen parte de esos escenarios que a hoy tienen más necesidades de las que pueden ser cubiertas por los profesionales que se están generando en las universidades. *¿Será que se ha convertido en apremiante educar a la población en general en las artes y la ciencia de construir software?*

Entre tanto las solicitudes siguen en aumento, y como ya lo he mencionado, cada día hay más necesidades en las empresas de hacer software del tipo que sostiene los procesos de negocio importantes, y mientras muchos intentamos motivar a los jóvenes en formación sobre lo edificante de ejercer profesionalmente el rol de ser constructor de software, el planeta se desborda en iniciativas en las que es bienvenido construir software de cualquier manera. *¿Desmotivar el trabajo serio y el ejercicio profesional será la salida correcta a la intensa demanda de software? ¿A qué costo se pretende figurar como empresa, marca o país en los indicadores de las tiendas de aplicaciones? ¿Dónde queda el ejercicio académico de hacer de la calidad un hábito de las personas dedicadas al software?*

Son muchos los actores implicados en el detrimento del ecosistema profesional y académico de las ingenierías de sistemas e informática, **no es el caso culpar el discurso de moda de semejante situación**, sin embargo si es un llamado de atención. La brecha entre el mundo académico y el empresarial está planteada hace mucho tiempo en más de un área de conocimiento, hoy, en el área de la construcción de software hay una brecha más, entre los escenarios que requieren de la profesionalización o al menos de una ejecución más sensata y centrada en la calidad y los esfuerzos e investigaciones de tantas personas, y los escenarios de los emprendedores y sus máquinas de hacer empresas y aplicaciones en búsqueda de alguna que sea exitosa. La pregunta final del planteamiento de este problema es *¿Cómo motivamos la profesionalización ante el discurso de marketing del emprendimiento moderno?*

Las propuestas

Hace un tiempo escuché hablar a un empresario local, cuya compañía es exitosa y reconocida en el campo de la ingeniería de software, por contarse entre las pocas del país que han alcanzado el nivel 5 de CMMI, de hecho por ser la primera. Sus ideas planteaban los niveles de insensatez que vivíamos y de cómo la ingeniería de software no servía para mucho más que para que tener proyectos igualmente fallidos, diría yo, fallidos con estilo y muy costosos. Al finalizar de hablar este hombre, después de echar por el piso la moral de todos los asistentes al congreso académico en el que participábamos, salió del salón sin ofrecer ni una opción, ni una esperanza, y lo peor, sin lugar a objeciones o debates.

Ante tal desfachatez me enojé bastante, sin embargo hoy al finalizar este artículo entiendo algo de eso que llamé “su propia insensatez”. Es insensato levantar todas estas quejas sin ofrecer una solución al respecto, lo es, y no saben cuánto lo lamento.

Considero sin embargo que hay personas que deseamos hacer cosas diferentes a lo que vemos que sucede en nuestras realidades laborales, y de ahí surgen cantidad de movimientos en la red y en la cotidianidad, de gente y comunidades dispuestas a enseñar lo que saben, a transmitir toda esa experiencia que adquieren en el camino e intentar que otros pasen por vías diferentes.

Quizá es un paliativo, pero es lo que podemos hacer quienes no tenemos el poder de movilizar gobiernos, instituciones y empresas a cambiar de mentalidad.

Levantar la voz, cuestionar a otros sobre lo que está sucediendo, incentivar a otros a compartir y seguir intentando cambiar el rumbo, son acciones alcanzables y tangibles, por lo que continuar haciendo esfuerzos de este tipo desde todas las iniciativas colectivas e individuales posibles, propende captar la atención de aquellos que en su labor si tienen *el poder y la responsabilidad* de impactar positivamente los escenarios en los cuales se le otorgue un mayor nivel de dignidad al esfuerzo por **la profesionalización y ejercicio digno de la Ingeniería de Software** como aporte a los distintos ámbitos sociales.

A título personal: académico y empresarial

Escribir este artículo tiene varias razones de ser, quizá demasiado extensas para ser publicadas, sin embargo un par de ellas son importantes. La situación es real, **es real en los entornos académicos y empresariales**, espacios en los cuales me desenvuelvo como profesional actualmente y donde hay muchas más cosas que decir que las que podía escribir acerca de cómo estos entornos están los principales *causas y causantes* de las desmotivaciones actuales por el estudio universitario de este tipo de profesiones.

Sin embargo, han existido y existimos los que creemos que aunque el ecosistema académico requiere muchos ajustes, este tipo de formación te da herramientas que enriquecen tu visión y ejercicio profesional, desde el estudio disciplinado, hasta la formación en ética y sensibilización de las situaciones sociales y humanas, causas por las cuales se forman todo tipo de profesionales.

A título personal, esto no me hace ajena a saber que hay grandes talentos, admirados por muchos en el ámbito de la tecnología y hasta en la cotidianidad, **que no han necesitado de formación académica**, pero creo que la problemática social va un poco más allá de las *excepciones* y llega al punto de hablar de las oportunidades que tiene la población general y de hecho las necesidades de la misma.

Es en esa óptica espero la lectura de este artículo, pues si bien soy del tipo de persona que promueve el aprendizaje, el autoestudio en todas sus formas y maneras, considero que el ejercicio de desmeritar la academia bajo la premisa de que los *"héroes"* tecnológicos de nuestra generación jamás terminaron la universidad o de que muchos hoy en día afirmen incluso que sus hijos no asistirán a ella, **no es en absoluto un discurso positivo**, y que es necesario equilibrar los mensajes de las partes y como siempre, *cerrar brechas entre los diferentes entornos propuestos.*

Las cuentas claras y el proceso de desarrollo concreto

SOFTWARE QA

La calidad en los procesos de desarrollo de software es más que llenar documentos por llenar: es el esqueleto de soporte para la toma de decisiones y la asertividad en la satisfacción del cliente. No todo es camisa de fuerza, pero muchas de sus herramientas facilitan el trabajo y disminuyen la presencia de errores y/o problemas durante la ejecución del proyecto.

Escrito por: **Eliana Caraballo** (Ing. de sistemas)

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.

Redes sociales:
Twitter / Identi.ca: **@elianaca**
LinkedIn: http://co.linkedin.com/in/elianacaraballoa

Imagine este escenario: usted tiene hambre y entra a un restaurante. El mesero amablemente le pregunta qué desea comer, a lo que usted le responde: "algo rico". Seguidamente quien lo atiende le nombra todos los platos que hay en la carta y usted simplemente le dice: "pues tráigame lo mejor que tenga disponible". El mesero le describe el plato que para él es el mejor del restaurante pero usted le dice que no le gusta la carne y se retira argumentando que se va para otro restaurante donde entiendan lo que quiere. El mesero se queda pensando en por qué no indagó lo que el cliente quería y le gustaba antes de ofrecerle el plato. Si trasladamos este contexto al ambiente de desarrollo de proyectos de software: ¿Se le hace familiar? Muy seguramente sí.

Uno de los problemas más comunes en este tipo de proyectos es lograr concretar lo que hay que hacer para que el software funcione como debe de ser. Dice la máxima que el

cliente siempre sabe lo que quiere, lo que no tiene claro es cómo lo quiere, y es ahí donde si se tiene un buen soporte de procesos y los mecanismos necesarios para el mismo, la probabilidad de que el proyecto sea exitoso aumenta. Es importante siempre exigir claridad sobre lo que se va a hacer y nunca trabajar sobre supuestos; inclusive ayudarle a aterrizar las ideas y guiarlo en caso de que esté equivocado en ciertos conceptos técnicos, pues si las bases del proyecto están mal hechas, muy probablemente terminará siendo un "proyecto atípico" que consumirá recursos y generará pérdidas en lugar de ganancias.

El proceso de acercamiento y establecimiento de relaciones con el cliente puede ser visto como un matrimonio: primero hay que ser amigos y luego novios, antes de casarse con él. Como en toda relación, hay que descubrirlo, conocerlo, para poder así enamorarlo y amarrarlo. Si desde el comienzo se intentan forzar las relaciones comerciales simplemente por ser un "pez gordo", se caerá en el círculo de "cliente insatisfecho – sobrecostos para contentar al cliente" que no es sano para el ambiente propio de la empresa. Cuando esté en esta etapa, tenga presente la siguiente frase de mercadeo:

> *"Recuerde que el usuario sabe de usted, lo mismo que usted de él: ¡NADA!".*

Es importante aclarar que el concepto de calidad es bastante amplio y no todos sus aspectos van a ser tratados acá. Quiero enfocarme principalmente en la necesidad de un sistema de calidad acompañado de procesos CMMi en una casa desarrolladora. Pensar en esto hace que se nos venga una cosa a la cabeza: ¡Más documentos! ¡Más cosas por llenar! y si... contra la documentación exhaustiva de lo que codificamos no podemos hacer nada, pero si le damos verdadero sentido entenderemos el por qué es importante.

En lo primero que se debe pensar es en las reglas del juego: **la gestión de la configuración**. Cómo se van a establecer las líneas base, de qué manera se va a hacer seguimiento y control de los cambios requeridos y cómo voy a asegurar la integridad de lo que se está desarrollando. Implantar las buenas prácticas de nombrado, codificación, comentarios y demás permitirán que tanto la construcción como el soporte del producto sea mucho más sencilla y agradable para el equipo de trabajo. Una regla de oro en la programación es: "Codifica pensando en que vas a ser tú quien va a hacerle mantenimiento a la aplicación, así que escribe el código como te gustaría encontrarlo". Sé que es suena a cartel de baño pero es la realidad. Si todos codificamos con esta regla en mente, nos facilitaremos y le facilitaremos el trabajo a los que vienen detrás. No hay peor infierno que tener que modificar el código ajeno, y más si este no está lo suficientemente claro y no tiene documentos de apoyo para su entendimiento.

Una buena práctica es desarrollar pensando: ¿Y si mañana no estoy? Si todo está debidamente documentado y escrito de forma que cualquiera lo pueda entender, el mantenimiento y soporte al programa no será complicado. Sino, piense: ¿Qué sería del catolicismo sin la Biblia?: cualquier persona que quiera conocer acerca de ellos, de sus reglas y rituales, o simplemente su historia, debe referirse a esta sin falta. Igualmente debe ser para el proyecto el sistema de gestión de la configuración: un soporte al cuál

acudir en caso de dudas, o para simple conocimiento del mismo.

Lo segundo a considerar es **la gestión de los requerimientos**. El por qué hay que documentarlos es claro: el cliente cambia de parecer constantemente, y si no desarrollamos sobre una línea base clara y definida, nuestro proyecto será eterno y complejo, pues ellos siempre querrán agregar o cambiar funcionalidades al mismo costo ("¿Qué tanto es poner un botón adicional que llene un campito?"). Adicionalmente si no está escrito, es muy probable que vayamos olvidando ciertas cosas que se hablaron en su momento para el funcionamiento del software. En todo proceso de toma de requisitos es importante que el cliente apruebe y valide todos y cada uno de ellos, ya que eso evita confusiones sobre lo que hay qué hacer y cómo debe de funcionar al final.

En las etapas de análisis y diseño es crucial identificar cuáles artefactos de la ingeniería de software son los que se van a usar para cada requisito: no todos son indispensables para todos; todo depende de la complejidad del mismo. El prototipo, los diagramas de secuencia, los diagramas de clase, los casos de uso, y todo cuanto se nos ocurra que hayamos visto en la universidad son los que a nosotros, como desarrolladores, nos darán idea de lo que tendremos que codificar (lo que el cliente quiere). Es nuestra maqueta, nuestro plano, nuestro esquema. No imagino a un ingeniero civil comenzar a construir sin tener planos del lugar, o a un cirujano cortar sin haber trazado el mapa de cortes. Es trabajo de los gerentes de software, los arquitectos y los analistas de requisitos hacerle entender al cliente la importancia de estos documentos, y que no es simplemente "tiempo" que mal usado en lugar de comenzar a trabajar.

Ya con estas bases claras, es mucho más sencillo y agradable hacer la planificación, estimaciones y establecer los compromisos para que sea un proyecto exitoso. A partir de este punto, la dinámica que se genere entre el gerente y el equipo de desarrollo hará que la “máquina” de desarrollo ande perfectamente engranada, y que el “proceso de producción” funcione como debe de ser.

Aparte de la gestión de desarrollo como tal, es importante que el gerente defina para cada proyecto la forma en que se va a hacer seguimiento y control y establecer las acciones correctivas en caso de ser necesario. Adicionalmente, este debe establecer métricas claras. Estos dos temas, al ser más de gerencia que de desarrollo, serán tratados en un artículo posterior.

Mi invitación es hoy a dignificar nuestro trabajo de desarrollo y darle su verdadero valor. El proceso de desarrollo sin la ingeniería de software es caótico, desordenado e impredecible; y es nuestro deber aprender a integrarla cuando codificamos y hacerla parte de nuestro diario ejercicio. La calidad no es un concepto nuevo, o una moda, y entre más la tengamos presente, mejor será nuestro trabajo y mayor nuestra productividad. En palabras de mi papá:

> *"No está mal ser pegador de ladrillos, lo que está mal es no ser EL MEJOR (o al menos intentarlo)".*

La Web Semántica y sus Ontologías

INTELIGENCIA ARTIFICIAL

La Web Semántica es una extensión de la Web convencional que permitirá encontrar, compartir y combinar la información más fácilmente.

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**

La Web semántica es un área nacida de la unión de la Inteligencia Artificial y las Tecnologías Web, la cual propone nuevas técnicas y paradigmas para la representación de la información que facilite la compartición e integración de recursos a través de la Web. Sus principios básicos son la descentralización, compartición, compatibilidad y la apertura al crecimiento; y se apoya de un conjunto de definiciones de conceptos llamadas Ontologías.

Si bien HTML ha tenido una evolución a lo largo de los años como es el caso del actual HTML5 aún no se posee la cantidad de metadatos que permita a los buscadores responder preguntas directas como "localiza todos los archivos relacionados con la investigación del Genoma Humano", donde la Web Semántica tendrá el papel de modificar la manera en que se busca y presenta la información en Internet.

Objetivo

Su principal objetivo es lograr una estandarización de todos sus datos (audio, servicios, imágenes, etc), presentándose en un mismo formato el cual pueda ser leído por Agentes Inteligentes que clasificarían los datos de una manera eficiente, devolviendo resultados más precisos ante las peticiones de los usuarios.

Este enfoque propone que la Web sea enriquecida de tal forma que los datos sean reestructurados y se adicionen componentes semánticos que sean procesados de manera automática, para ello será necesario el uso de las Ontologías que posibilitan que los datos puedan ser utilizados por los Agentes Inteligentes de manera que hagan inferencias automáticamente localizando otros enlaces relacionados con sus metadatos, de tal forma que la respuesta que recibe el usuario sea más completa.

Resource Description Framework

RDF (Resource Description Framework) es uno de los lenguajes propuestos para ser utilizados en la Web Semántica, y permite añadir significado a los Sitios Web y así tener mayor cantidad de datos en cada enlace.

Por ejemplo, si visitáramos la página oficial de Hackers and Developers nuestro diagrama estructural con RDF sería el siguiente:

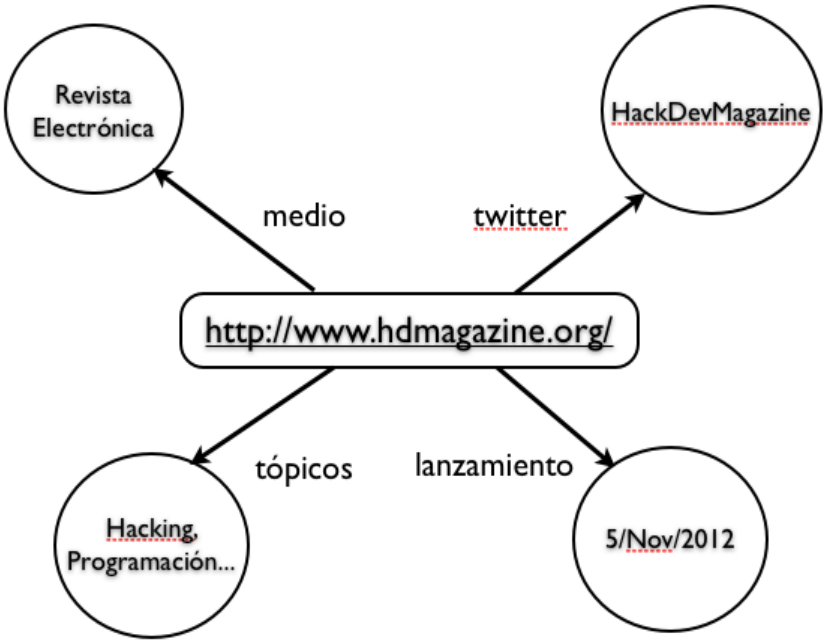


Diagrama Estructural con RDF

En él se indica que la dirección http://www.hdmagazine.org es una revista electrónica con fecha de publicación del 5 de Noviembre de 2012, donde sus tópicos son Hacking, Programación, etc y su cuenta de Twitter es HackDevMagazine. Como pueden notar va más allá de indicar en los metatags palabras clave y de esa manera encontrar más enlaces relacionados con esos metadados indicados.

Lo mostrado en la gráfica es expresado en el lenguaje RDF con una estructura similar a XML, se presenta a continuación:

```
<rdf:RDF>
      <rdf:Description rdf:about="http://www.hdmagazine.org">
            <h:Medio>Revista Electrónica</h:Medio>
            <h:Twitter>HackDevMagazine</h:Twitter>
            <h:Topicos>Hacking, Programación</h:Topicos>
            <h:Lanzamiento>5/Nov/2012</h:Lanzamiento>
      </rdf:Description>
</rdf:RDF>
```

Así los Agentes podrán analizar el código anterior y comparar con otros enlaces si puede encontrar relación entre los elementos y responder con enlaces más completos y se relacionen con HackDevMagazine.

Buscadores

Actualmente existen buscadores que están empezando a integrar la Web Semántica, y aunque Google se acerca a uno de ellos aún no hace el salto por completo, sin embargo, los buscadores que te recomiendo visitar son:

- WolframAlpha. http://www.wolframalpha.com/

- Quintura. http://quintura.com/

- Cluuz. http://cluuz.com/

- Swoogle. http://swoogle.umbc.edu/

En especial WolframAlpha y Quintura tienen una forma muy particular de presentar sus resultados, por ejemplo, agruparlos en categorías o presentar una nube de palabras clave en caso de que aún no sepas que deseas buscar.

Se pretende que la Web Semántica proporcione un avance cualitativo sobre el potencial de la Web actual, y como objetivo poder unificar los datos existentes; es por eso que se busca una estructura con la suficiente capacidad expresiva y de representación como es el caso de RDF.

Todo es cuestión de tiempo y estándares, y así como se hizo la evolución de una Web 1.0 a 2.0, lo mismo sucederá con la integración de la Web Semántica a los buscadores junto con la Inteligencia Artificial, Aprendizaje Automático, entre otros, logrando así el inicio de una nueva era llamada Web 3.0.

La Web Semántica se utiliza a veces como sinónimo de "Web 3.0" -Tim Berners-Lee

ASCII ART

Butterfly

por **chris.com**

TU ZONA EXCLUSIVA

U!

U! *(onomatopeya del término inglés «**you**»)* es la **zona exclusiva para lectores** de Hackers & Developers Magazine. Si quieres aparecer en esta sección, **envíanos un correo electrónico a lectores@hdmagazine.org** o un Tweet con el **hashtag oficial #HDMagazine** e incluiremos tu carta o comentario en el **U!** del próximo número.

En el próximo número de HD Magazine, me gustaría...

"(…) que se hablara sobre Ética Hacker o algo por el estilo, sobre top distros GNU-Linux, y algo sobre NOSQL en la actualidad" por **David Guerrero Morales** vía Google+

"(…) leer sobre medidas de seguridad en cuanto a la protección de Datos Personales para evitar el robo de identidad en la red" por **Benito Roque** vía Google+

"(…) que se hablase de análisis de logs en servidores web; un tema muy amplio, pero también muy interesante" por **David Gutierrez** vía Google+

"(…) ver el tema de las sesiones vs cookies, en cuanto a la seguridad y eficiencia que nos puede proporcionar cada una de ellas" por **Sandra Bautista Ortiz** vía Google+

"(...) sería genial un articulo «No recaer en las malas prácticas al usar frameworks php» pues siempre me pasa que cuando uso uno , me termina complicando y enredando recurriendo a las antiguas técnicas de malas prácticas de PHP" por **Luis Aguilera Clarke** vía Google+

Empleo ofrecido:

Si tienes una empresa, estudio y/o consultora de empleo y buscas personal IT en el área de desarrollo, administración de sistemas o liderazgo de proyectos, envíanos tu oferta de empleo a **contacto@hdmagazine.org**.

Publicación de ofertas de empleo gratuita, sujeta a disponibilidad.