	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**
CICLO I-2021	**Lenguajes Interpretados en el Servidor** **Guía de práctica No. 7** **Expresiones Regulares**

I. OBJETIVOS

- Entender qué es una expresión regular y el uso que puede darles para resolver distintos problemas.
- Construir patrones de expresiones regulares con distintos propósitos, como validación, búsqueda y reemplazo en cadenas.
- Utilizar las funciones de PHP diseñadas para trabajar con expresiones regulares.

II. INTRODUCCIÓN TEÓRICA

¿Qué son las expresiones regulares?

Una expresión regular puede ser considerada como una secuencia o patrón de caracteres que representa o describe a un conjunto de cadenas de caracteres sin enumerar explícitamente sus elementos. Estos patrones son utilizados para ser comparados contra una cadena de texto sobre la cual se necesita hacer una búsqueda para encontrar posibles coincidencias. Buena cantidad de lenguajes de programación utilizan las expresiones regulares porque poseen una indiscutible potencia para procesar texto cuando se intenta localizar caracteres que se ajusten a un formato o para hacer sustitución de cadenas dentro de un texto.

La utilización que se le da a las expresiones regulares suelen ser:

1. Verificación de datos que son enviados desde un formulario antes de insertarlos en una base de datos.

2. Localización de subcadenas dentro de una cadena de texto más compleja.
3. Búsqueda y reemplazo de secuencias de caracteres en una cadena de varias líneas, permitiendo incluso múltiples reemplazos.

Por qué utilizar expresiones regulares

Es muy probable que logre hacer con funciones de cadena y con un ciclo o lazo el mismo trabajo que con una expresión regular; sin embargo, cuando se hace uso de expresiones regulares se realiza código más compacto e intuitivo cuando se necesita hacer uso de coincidencia de patrones para realizar alguna tarea con

cadenas de texto. Además de esto, la ejecución de una expresión regular será mucho más rápida que hacer uso de las funciones de cadena y los ciclos o lazos.

Conceptos generales de expresiones regulares

Comencemos por definir lo que es un **patrón**, que indirectamente viene siendo un sinónimo de expresión regular. Un **patrón** es el modelo que se define para ser comparado contra un conjunto de cadenas de caracteres. Ese modelo puede estar constituido por **caracteres normales** (o literales) y por **metacaracteres**, que no son otra cosa más que caracteres con significado especial.

Por ejemplo, en la expresión regular "^1*a$" (las comillas no son parte de la expresión regular, solo la delimitan) el "1" y la "a" son caracteres normales o literales, mientras que los símbolos: "^", "*" y "$" son **metacaracteres** o caracteres con un significado especial que permiten que esta expresión regular se pueda comparar contra muchas otras cadenas en busca de coincidencias.

Dentro de los patrones que representan a un único carácter, podemos distinguir dos tipos:

- Los caracteres que se representan a sí mismos.
- Las categorías o clases de caracteres.

Agrupamiento de patrones

Los patrones que representan un carácter se pueden agrupar para formar expresiones regulares más complejas

y es esta característica las que les da una gran potencia para el procesamiento de cadenas. Entre los distintos tipos de agrupamiento de patrones se pueden mencionar:

- Las secuencias.
- Los cuantificadores.
- Los paréntesis.
- Las alternativas.
- La fijación de patrones.
- Los caracteres escapados.
- La precedencia.
- Las operaciones.

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #7: Expresiones regulares	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP sublime Text o Eclipse PHP	1
4	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

Indicaciones: Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que el PHP Designer no es compilador solamente un editor. Por lo tanto, los errores de sintaxis los podrá observar únicamente hasta que se ejecute el script cargando la página en el navegador de su preferencia.

Ejemplo 1: El siguiente ejemplo muestra cómo utilizar expresiones regulares para contar las palabras de un texto ingresado en un área de texto. El script utiliza manejo de eventos con JavaScript no obstrusivo, de modo que no encontrará atributos HMTL como onclick, onmouseover, etc.

Archivo 1: contadorpalabras.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
     <!-- Indicar al navegador que el sitio estará optimizado para
dispositivos móviles -->
             <meta   name="viewport"   content="width=device-width,
initial-scale=1.0"/>
                         <meta      http-equiv="Content-Script-Type"
content="text/javascript" />
    <title>Contador de caracteres y palabras</title>
    <!-- Importar iconos de las fuentes de Google -->
    <!-- <link rel="stylesheet"
href="http://fonts.googleapis.com/icon?family=Material+Icons"     />
-->
    <!-- Incluyendo estilos de la librería materialize CSS -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.5/css
/materialize.min.css" />
    <link rel="stylesheet" href="css/palabras.css" />
</head>
<body>
<header class="row s12">
```

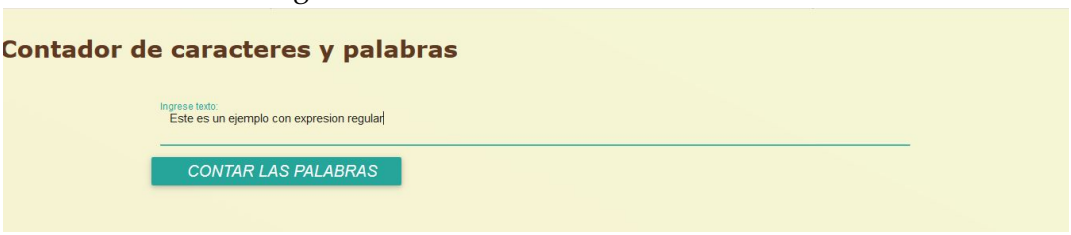
```
    <h1>Contador de caracteres y palabras</h1>
</header>
<section>
<article class="container">
<?php
if(isset($_POST['btnContar'])):
    //Se carga la librería con la función que se encargará
    //de contar las palabras en el texto ingresado por el usuario
    require "wordcount.php";
    $textolimpio = wordcount($_POST['area']);
    $complemento = count($textolimpio)>1 ? "palabras" : "palabra";
      echo "<p>Se han contabilizado: " . count($textolimpio) . "
$complemento</p>\n";
else:
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST"
name="frmtexto" id="frmtexto" class="col s12">
<div class="row col s12">
    <div class="input-field col s12">
                            <textarea   name="area"   id="text-box"
class="materialize-textarea col s12"></textarea>
        <label for="text-box">Ingrese texto:</label><br />
    </div>
    <div class="12u">
             <button type="submit" name="btnContar" id="btnContar"
class="btn waves-effect waves-light">
           <i class="material-icons right">Contar las palabras</i>
        </button>
    </div>
</div>
</form>
<script src="js/palabras.js"></script>
<?php
endif;
?>
</article>
</section>
</body>
<!--[if lt IE 9]>
<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.js"></scr
ipt>
<script
src="http://ajax.googleapis.com/ajax/libs/jqueryui/1/jquery-ui.min.
js"></script>
<!-- Importando librería de JavaScript de materialize CSS -->
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.5/js/m
aterialize.min.js"></script>
</html>
```

Archivo 2: wordcount.php

```
<?php
//El texto recibido en $texto es el texto que
//se ingresó en la caja de texto del formulario
function wordcount($texto){
    //Eliminando espacios en blanco entre palabras
    $textoarea = preg_replace("/\s+/u"," ", trim($texto));

    //Contando las palabras separándolas por el espacio en blanco
    //y almacenándolas en la matriz $palabras haciendo uso de la
    //función preg_split()
    $palabras = preg_split("/\s/u", $textoarea);
    return $palabras;
}
?>
```

El resultado en el navegador seria:



Ejemplo #2: El siguiente ejemplo muestra como listar los enlaces de una web determinada mediante su dirección web, haciendo uso de coincidencias en base a expresiones regulares. Para comenzar ingrese una url en la caja de texto superior derecha, el proceso puede tardar dependiendo de la cantidad de enlaces encontrados.

Archivo 1: listaenlaces.php

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Ejercicio de Expresiones Regulares</title>
<link rel="stylesheet" href="css/styles.css" />
<!--[if IE]>
<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

```
<!--[if IE 6]>
<script src="js/belatedPNG.js"></script>
<script>
    DD_belatedPNG.fix('*');
</script>
<![endif]-->
</head>
<body>
<div id="bodywrap">
<section id="pagetop"></section>
<header id="pageheader">
    <h1>Uso de<span> Expresiones Regulares</span></h1>
    <div id="search">
        <form action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="post" name="frm" id="frm">
        <div class="searchfield">
            <input type="text" name="url" id="url"
value="http://www.catswhocode.com/blog/15-php-regular-expressions-for
-web-developers" required="required">
        </div>
        <div class="searchbtn">
            <input type="hidden" name="Enviar" value="Enviar" />
            <input type="image" src="images/searchbtn.png"
alt="search">
        </div>
        </form>
    </div>
</header>
<div id="contents">
<section id="main">
<div id="leftcontainer">
<article class="post">
    <h2>Listar enlaces de una web</h2>
    <p>El siguiente ejemplo muestra como listar los enlaces
    de una web determinada mediante su direccion web, haciendo
    uso de coincidencias en base a expresiones regulares.</p>
    <p>Para comenzar ingrese una url en la caja de texto
    superior derecha, el proceso puede tardar dependiendo
    de la cantidad de enlaces encontrados</p>
    <?php
        if(isset($_POST['Enviar'])){
            procesarForm();
        }
        else{
            echo "<div style='height:300px;'></div>";
        }

        function procesarForm(){
            $url = isset($_POST['url']) ? $_POST['url'] : null;
            //Expresión regular para encontrar enlaces dentro de
            //un sitio web usando la función file_get_contents de PHP
```

```
            if(!preg_match('|^http(s)?\://|', $url)){
                $url = "http://$url";
            }
            $html = file_get_contents($url);
            preg_match_all("/<a\s*href=['\"](.+?)['\"].*?>/i", $html,
$matches);
            echo "<div style='clear:both;'></div>";
            echo "<h2>Enlaces encontrados en ".
htmlspecialchars($url) .": </h2>";
            echo "<ul>";
            for($i=0; $i < count($matches[1]); $i++){
                echo
"<li>".htmlspecialchars($matches[1][$i])."</li>";
            }
            echo "</ul>";
        }
    ?>
<div class="clear"></div>
</article>
</div>
</section>
<div class="clear"></div>
</div>

</div>
<footer id="pagefooter">
<div id="footerwrap">
</footer>
</body>
</html>
```

El archive de estilos se proporcionará con los recursos de la guía de práctica #5 que podrá descargar desde el sitio web de la Universidad Don Bosco (http://www.udb.edu.sv) en la sección guías de práctica.

El resultado si ejecuta el script en el navegador de su preferencia debería ser algo como lo siguiente:

Ejercicio #3: El siguiente ejemplo permite buscar una palabra dentro de un texto ingresado en un área de texto. Puede copiar y pegar en esta área de texto contenido de alguna página web y verificar ingresando en el cuadro de texto superior la palabra a buscar. Al presionar el botón de búsqueda de palabra se marcarán todas las ocurrencias de las palabras encontradas.

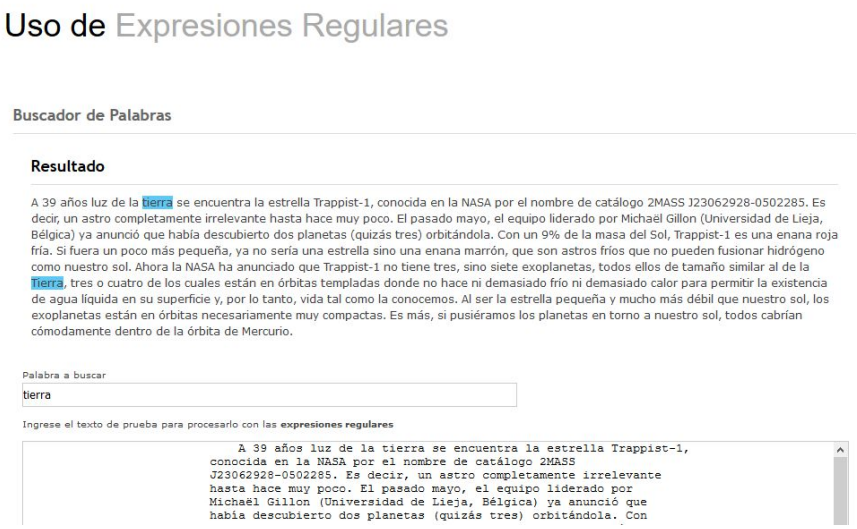
Archivo 1: buscadorpalabras.php

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Ejercicio de Expresiones Regulares</title>
<link rel="stylesheet" href="css/styles.css" />
<!--[if IE]>
<script
      src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></scr
      ipt>
<![endif]-->
<!--[if IE 6]>
<script src="js/belatedPNG.js"></script>
<script>
    DD_belatedPNG.fix('*');
</script>
<![endif]-->
</head>
<body>
<div id="bodywrap">
<section id="pagetop"></section>
<header id="pageheader">
    <h1>Uso de<span> Expresiones Regulares</span></h1>
</header>
<div id="contents">
    <section id="main">
        <div id="leftcontainer">
```

```
            <h2>Buscador de Palabras</h2>
            <section id="sidebar">
            <?php
              if(isset($_POST['Enviar'])){
                  $text = isset($_POST['comment']) ?
        trim($_POST['comment']) : null;
                  $palabra = isset($_POST['palabra']) ?
        trim($_POST['palabra']) : null;
                  $text = preg_replace("/\b(".$palabra.")\b/i",
        '<span
    style="background:#5fc9f6">\1</span>', $text);
            ?>
            <div id="sidebarwrap">
                <h2>Resultado</h2>
                <p><?=$text?></p>
            </div>
            <?php
              }
            ?>
          </section>
          <div class="clear"></div>
          <article class="post">
          <form action="<?php echo $_SERVER['PHP_SELF'] ?>"
        method="post" class="form">
                  <p class="textfield">
                      <label for="palabra">
                          <small>Palabra a buscar</small>
                      </label>
                      <input name="palabra" id="palabra"
        value="tierra" size="22" tabindex="1" type="text" />
                  </p>
                  <p>
                      <small>Ingrese el texto de prueba para
        procesarlo con las
                          <strong>expresiones regulares</strong>
                      </small>
                  </p>
                  <p class="text-area">
                      <textarea name="comment" id="comment" cols="50"
        rows="10" tabindex="4">
                          A 39 años luz de la tierra se encuentra la
        estrella Trappist-1,
                          conocida en la NASA por el nombre de
        catálogo 2MASS
                          J23062928-0502285. Es decir, un astro
        completamente irrelevante
                          hasta hace muy poco. El pasado mayo, el
        equipo liderado por
                          Michaël Gillon (Universidad de Lieja,
        Bélgica) ya anunció que
```

```
                    había descubierto dos planetas (quizás
        tres) orbitándola. Con
                    un 9% de la masa del Sol, Trappist-1 es una
        enana roja fría.
                    Si fuera un poco más pequeña, ya no sería
        una estrella sino
                    una enana marrón, que son astros fríos que
        no pueden fusionar
                    hidrógeno como nuestro sol.
                    Ahora la NASA ha anunciado que Trappist-1
        no tiene tres, sino
                    siete exoplanetas, todos ellos de tamaño
        similar al de la Tierra,
                    tres o cuatro de los cuales están en
        órbitas templadas donde
                    no hace ni demasiado frío ni demasiado
        calor para permitir la
                    existencia de agua líquida en su superficie
        y, por lo tanto,
                    vida tal como la conocemos. Al ser la
        estrella pequeña y mucho
                    más débil que nuestro sol, los exoplanetas
        están en órbitas
                    necesariamente muy compactas. Es más, si
        pusiéramos los planetas
                    en torno a nuestro sol, todos cabrían
        cómodamente dentro de
                    la órbita de Mercurio.
                </textarea>
            </p>
            <p>
                <input name="Enviar" id="Enviar" value="1"
        type="hidden" />
                <input name="submit" id="submit" tabindex="5"
        type="image" src="images/submit.png" />
            </p>
            <div class="clear"></div>
        </form>
          <div class="clear"></div>
        </article>
      </div>
    </section>
    <div class="clear"></div>
    </div>
</div>
<footer id="pagefooter">
    <div id="footerwrap">
    </div>
</footer>
</body>
</html>
```

El resultado en el navegador es el siguiente:



Ejercicio #5: El siguiente ejemplo muestra cómo utilizar una expresión regular para determinar si uno o varios archivos que se desean enviar al servidor son archivos de imagen válidos para publicar en sitios web o no. Se utiliza un control de formulario input type=file para adjuntar uno o varios archivos y en una secuencia de comando o guión PHP se procesan los nombres de los archivos enviados para determinar de acuerdo a su extensión si son o no archivos de imagen.

Archivo 1: uploadfile.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <!-- Indicar al navegador que la página estará optimizada para
      distintos dispositivos -->
    <meta name="viewport" content="width=device-width,
      initial-scale=1.0"/>
    <!--Import Google Icon Font-->
    <link rel="stylesheet"
      href="http://fonts.googleapis.com/icon?family=Material+Icons"
      />
    <link rel="stylesheet" href="css/fonts.css" />
    <!--Import materialize.css-->
    <link rel="stylesheet" href="css/materialize.css" />
</head>
<body>
<section>
<article>
<div class="row">
```

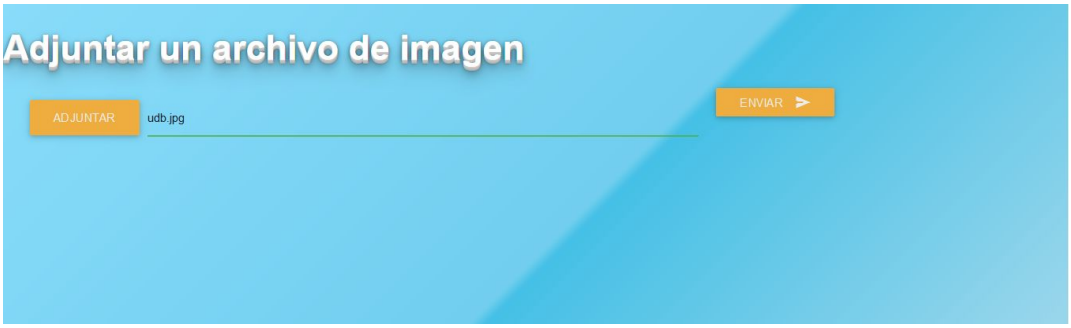
```
    <h1 class="title-form">Adjuntar un archivo de imagen</h1>
    <?php
      if(isset($_POST['send'])):
          //Incluir librería de funciones
          require_once("comprobarimagen.php");
          //Verificar si se han enviado uno o varios archivos
          //valiéndonos de una expresión regular
          $archivos = array();
          if(!empty($_FILES['files']['name'][0])):
              $list = "<ol class=\"list-files\">\n";

              foreach($_FILES['files']['name'] as $i => $archivo):
                  $archivos[$i] = $archivo;
                  //Invocar a la función que verificará mediante
                  //expresión regular si el archivo pasado como
                  //argumento es o no es imagen.
                  $list .= "<li>\n<a href=\"#\">" . $archivos[$i] .
      comprobarimagen($archivos[$i]) . "</a>\n\t</li>\n";
              endforeach;
              $list .= "</ol>\n";
              echo $list;
          endif;

          //Obteniendo los datos del formulario

      else:
    ?>
    <form action="<?=$_SERVER['PHP_SELF']; ?>" method="POST"
      enctype="multipart/form-data" class="col s12">
        <div class="row col s12">
            <div class="file-field input-field col s8">
                <div class="btn">
                    <span>Adjuntar</span>
                    <input type="hidden" name="MAX_FILE_SIZE"
      value="2097152" />
                    <input type="file" name="files[]"
      multiple="multiple" />
                </div>
                <div class="file-path-wrapper">
                    <input type="text" class="file-path validate"
      placeholder="Seleccione
sólo archivos de imagen" />
                </div>
            </div>
            <div class="row col s4">
                <button type="submit" class="btn waves-effect
      waves-light"
name="send">Enviar
                    <i class="material-icons right">send</i>
                </button>
            </div>
```

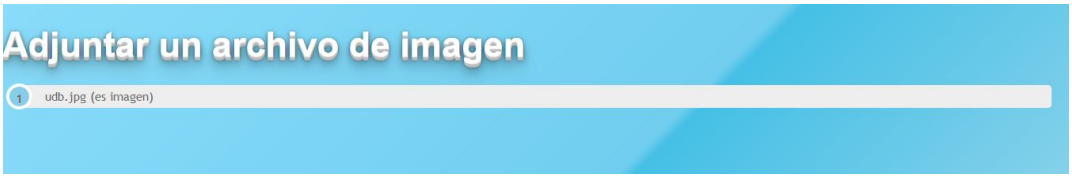
```
        </div>
    </form>
    <?php
        endif;
    ?>
</div>
</article>
</section>
</body>
<!-- Import jQuery before materialize.js -->
<script src="//code.jquery.com/jquery-2.1.1.min.js"></script>
<script src="js/materialize.min.js"></script>
</html>
```

Archivo 2: comprobarimagen.php

```
<?php
  function comprobarimagen($archivo){
      //La expresión regular analiza si el archivo es de
      //una extensión válida para una imagen gif|jpeg|jpg|png
      //utilizando la función preg_match()
      $patron = "/\.(gif|jpe?g|png)$/i";
      $verificado = preg_match($patron, $archivo);
      $esimagen = $verificado == true ? " (es imagen)" : " (no es
      imagen)";
      return $esimagen;
  }
?>
```

El resultado en pantalla seria:





V. DISCUSIÓN DE RESULTADOS

1. En el ejemplo 1 del procedimiento de esta guía de práctica, modifique el código para que cuando se ingresen signos de puntuación separados de las palabras tanto antes como después del signo, estos puntos no sean

contabilizados como palabras; es decir, que el conteo siga siendo únicamente de palabras con la expresión regular que se ha utilizado. Puede modificar esa expresión regular o utilizar otra más para poder controlar y/o posiblemente reemplazar la cadena para que los signos de puntuación queden justo a la par de la palabra que le precede y así no sean contados como palabras.

2. En el ejemplo 3 del procedimiento de esta guía de la práctica, modifique el código para que justo a la par de Resultado una vez que se envíe el formulario con la palabra que se desea buscar en el texto ingresado en el campo textarea, se indique el número de coincidencias que fueron encontradas de la palabra buscada en el texto ingresado. Se encontraron 2 coincidencias.

VI. BIBLIOGRAFIA

- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- http://www.php.net/manual/en