**JAVA-PROGRAMADOR**



**EXPRESIONES REGULARES**

ALUMNO:

Quesada Marcelo, Rubén Kevin Jean Pierre 20144524A

PROFESOR:

Tafur Anzualdo, Gelacio Albino

Lima, 19 de mayo del 2016 **ALUMNO:**

Quesada Marcelo, Rubén Kevin Jean Pierre 20144524A

**ALUMNOS:**

Meléndez Malca, Mayra

Felices Sanabria Lisbeth

**PROFESOR:**

Eric Gustavo, Coronel Castillo

Lima, 7 de marzo del 2020

# DEDICATORIA

Dedicamos y agradecemos al docente Eric Gustavo, Coronel Castillo por conocimientos brindados durante todo el tiempo que nos llevó a completar la culminación de este curso y en este trabajo de investigación. La inspiración para hacer la investigación provino de nuestra familia que es pilar fundamental en nuestra formación. Escribir ésta monografía ha sido una experiencia importante y formativa en nuestra vida.

Contenido

[DEDICATORIA 2](#_Toc34478570)

[RESUMEN 4](#_Toc34478571)

[DESARROLLO DEL TEMA 5](#_Toc34478572)

[Expresiones regulares en programación: Java 5](#_Toc34478573)

[Tipos de expresiones regulares en JavaScript 5](#_Toc34478574)

[Algunos elementos esenciales para hacer expresiones regulares en JavaScript: 6](#_Toc34478575)

[Casos concretos de uso de expresiones regulares 6](#_Toc34478576)

[CONCLUSIONES 9](#_Toc34478577)

[RECOMENDACIONES 9](#_Toc34478578)

## RESUMEN

En el presente trabajo de investigación abordaremos el tema de Expresiones Regulares y su aplicación en Java ya que las expresiones regulares proporcionan un método eficaz y flexible para procesar texto. La notación extensiva de búsqueda de patrones coincidentes de las expresiones regulares permite analizar rápidamente grandes cantidades de texto para buscar patrones de caracteres específicos; para validar un texto con el fin de asegurar que se corresponde con un patrón predefinido (por ejemplo, una dirección de correo electrónico); para extraer, editar, reemplazar o eliminar subcadenas de texto; y para agregar las cadenas extraídas a una colección con el fin de generar un informe. Para muchas aplicaciones que usan cadenas o analizan grandes bloques de texto, las expresiones regulares son una herramienta indispensable. Es por ello que nos enfocamos en la realización de un formulario de datos de Facebook para que así al validar los datos de una persona esta pueda acceder a crear su cuenta Facebook

# DESARROLLO DEL TEMA

## Expresiones regulares en programación: Java

Aprender a programar significa, entre muchos otros detalles, memorizar expresiones y pasos que permiten ejecutar acciones simples y prácticas. Se podría alcanzar un símil comprensible con los típicos atajos de teclado: con una sencilla combinación de teclas uno es capaz de acceder al explorador, cerrar el navegador, buscar un programa. En algunos lenguajes de programación, las expresiones regulares sirven para detectar patrones en distintas cadenas de texto. Para los programadores, esas expresiones regulares facilitan su labor en un solo paso.

Un ejemplo muy sencillo es la comprobación vía expresión regular que los datos rellenados por los usuarios en un formulario son correctos o no. Alguien puede utilizar un breve formulario para recabar el nombre, los apellidos y el teléfono de contacto de sus clientes, y que algunos de ellos no cumplimenten bien este último campo. La forma rápida de comprobarlo es a través de una expresión regular.

Los profesionales JavaScript pueden disfrutar de estas expresiones regulares, concretamente para hacer comprobaciones determinadas en las distintas cadenas de texto a partir de un patrón que se puede resumir en una expresión abierta.

## Tipos de expresiones regulares en JavaScript

Una expresión regular usa caracteres y meta-caracteres para definir, de forma abierta, patrones concretos en cadenas de texto. Esos caracteres, combinados unos con otros de forma especial, permiten extraer patrones o elementos concretos de esa cadena para buscar o manipular el texto (identificadores, correos electrónicos). En el caso de JavaScript, hay dos tipos de expresiones regulares:

● **EXPRESIONES REGULARES MEDIANTE UNA CADENA LITERAL PARA ENCONTRAR UN PATRÓN**

Para ello se usa el constructor de objeto ***RegExp***. Para crear un literal ***RegExp*** se utiliza esta expresión: ***var re = /regular expression/; .*** El patrón de expresión regular se encuentra entre la apertura y el cierre de las barras diagonales, una estructura que suele ser habitual y obligatoria.

● **EXPRESIONES REGULARES DE APLICACIÓN EN VARIAS CADENA**

***expReg.test(cadena).*** Si existe coincidencia, se devuelve true; y si no existe una coincidencia ente la expresión y un posible patrón, se devuelve false.

## Algunos elementos esenciales para hacer expresiones regulares en JavaScript:

* ^: el emparejamiento se debe realizar desde el principio de la cadena.
* [A-Z]: cualquier carácter entre la A mayúscula y la Z mayúscula.
* {1,2}: uno o dos caracteres.
* \s: un espacio en blanco.
* \d: un dígito.
* {4}: cuatro dígitos.
* \s: un espacio en blanco.
* ([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]): cualquier carácter entre la B mayúscula y la Z mayúscula, excepto las vocales.
* {3}: tres caracteres.
* $: el emparejamiento se debe realizar hasta el final de la cadena.

## Casos concretos de uso de expresiones regulares

Es habitual usar expresiones regulares para detectar si existe una secuencia concreta de letras o palabras o bien números en una cadena determinada. Para conseguir eso sólo es necesario utilizar una expresión regular simple:

* ***var reg = /javascript/;***  "Esto es lenguaje javaScript".match(reg);

// devuelve un array de un elemento ["javascript"] porque dentro de la cadena se encuentra un elemento con esa secuencia concreta de letras. En el caso de que no existiera las palabras “javaScript” dentro de la cadena, devolvería un elemento null.

* A veces se utiliza el elemento if para buscar coincidencias true o false a partir de una  
  combinación de caracteres y palabras (expresiones) en una cadena completa:

if ("Esto es lenguaje javaScript".match(/javascript/) {

// Se devolverá true porque si existe el elemento javaScript en la cadena.

}

if ("Esto es lenguaje Objective-C".match(/javascript/) {

// Se devuelve false porque no existe el elemento javaScript en la cadena.

}

Usando el elemento vinculado a la búsqueda de dígitos, las expresiones regulares se pueden utilizar para comprobar si una cadena tiene algún digito concreto, incluso programar expresiones regulares para detectar combinaciones o también repeticiones de una serie de dígitos dentro de una o varias cadenas de texto:

* Búsqueda de un dígito concreto en una cadena de texto: en este caso especial el programador busca si la cadena de texto contiene el número 2. Como sí lo contiene, la expresión regular devuelve lógicamente un ["2"].

"Este número es un 2".match(/\d/); // Devuelve un ["2"]

* Las expresiones regulares son capaces también de detectar la repetición de uno o varios dígitos dentro de una cadena de texto o una secuencia concreta de números dentro de esa misma cadena de texto. Lo único que es necesario es repetir dentro de la expresión el elementos \d, que debe ir acompañado, tanto en la apertura como el cierre, por el símbolo /:

"Este número es el 242".match(/\d\d\d/); // Devuelve el ["242"] porque dentro de esa cadena existe una expresión que contiene 3 dígitos seguidos.

* Para la búsqueda de dígitos en distintas cadenas de texto:

/\d{3}/ Busca 3 dígitos en la cadena  
/\d{1,5}/ Busca entre 1 y 5 dígitos en la cadena.  
/\d{2,}/ Busca 2 dígitos o más en la cadena.

Aquí algunos ejemplos concretos con esta última expresión regular: en el primer ejemplo, la expresión permite buscar dentro de una cadena de texto concreta (en este caso, “1234”), un máximo de dos dígitos, por eso devuelve el elemento ["12"]; en el segundo caso se solicita buscar en la cadena de texto entre uno y tres dígitos, por eso devuelve ["123"]; y en el tercer caso, la expresión está diseñada para buscar dentro de la cadena entre tres y diez dígitos, por eso devuelve cuatro, la cadena de texto completa: ["1234"].

"1234".match(/\d{2}/);

["12"]

"1234".match(/\d{1,3}/);

["123"]

1234".match(/\d{3,10}/)

["1234"]

# CONCLUSIONES

No hay una solución única en cada caso, nuestro método personal de resolver las expresiones complejas es mediante la técnica de divide y vencerás.

Hay un detalle a tener en cuenta en Java, como el carácter de escape es la «\», cuando la utilicemos en una cadena, tendremos que duplicarla, convirtiéndose por ejemplo \w en \\w.

# RECOMENDACIONES

Las expresiones regulares son un mecanismo de programación muy potente, pero esa potencia origina que sean complejas de utilizar. Comienza a trabajar con expresiones regulares sencillas y poco a poco intenta escribir expresiones más complejas.

Para ordenar los elementos de una lista puedes emplear diferentes algoritmos de ordenación