Máximo Hernández

ACCESO A DATOS -  RETO 0. EQUIPO ADKM

**Expresiones regulares**

Contenido

[Instrucciones: 1](#_Toc54287515)

[Descripción de las expresiones regulares 2](#_Toc54287516)

[El punto "." 2](#_Toc54287517)

[El signo de admiración "!" 2](#_Toc54287518)

[La barra inversa o contrabarra "" 2](#_Toc54287519)

[Los corchetes "[ ]" 3](#_Toc54287520)

[La barra "|" 4](#_Toc54287521)

[El signo de dólar "$" 4](#_Toc54287522)

[El acento circunflejo "^" 4](#_Toc54287523)

[Los paréntesis "()" 4](#_Toc54287524)

[El signo de interrogación "?" 5](#_Toc54287525)

[Las llaves "{}" 5](#_Toc54287526)

[El asterisco "\*" 6](#_Toc54287527)

[El signo de suma "+" 6](#_Toc54287528)

[Operadores de expresiones regulares 7](#_Toc54287529)

[Ejemplos de expresiones regulares 8](#_Toc54287530)

# Instrucciones:

**Sprint 4: Validación de los datos**

Se usan expresiones regulares de tal manera que se demuestra que se entiende su funcionamiento,

se gestionan las excepciones y se han realizado los teses.

(*Ander – 16/10/2020*):

1. Definir el alfabeto
2. Un alfabeto de dos elementos
3. Una longitud finita
4. Estructura en el código 🡪 2 while (evitar 3 whiles, que se ralentiza mucho)

(*Ander – 22/10/2020*):

1. Hay que permitir al usuario elegir el alfabeto.
2. Una expresión regular sin operadores debe definir toda la palabra, no coincidencias parciales
3. En caso que aparezcan más de 2 bucles anidados 🡪 utilizar recursividad
4. Operadores, empezar por el “\*”

# Descripción de las expresiones regulares

# *\*Origen info: wikipedia*

### El punto "."

El punto se interpreta por el motor de búsqueda como "cualquier carácter", es decir, busca cualquier carácter SIN incluir los saltos de línea. Los motores de Expresiones regulares tienen una opción de configuración que permite modificar este comportamiento. En .Net Framework se utiliza la opción RegexOptions.Singleline para especificar la opción de que busque todos los caracteres incluidos el salto de línea (\n).

El punto se utiliza de la siguiente forma: Si se le dice al motor de RegEx que busque "g.t" en la cadena "el gato de piedra en la gótica puerta de getisboro goot" el motor de búsqueda encontrará "gat", "gót" y por último "get". Nótese que el motor de búsqueda no encuentra "goot"; esto es porque el punto representa un solo carácter y únicamente uno. Si es necesario que el motor encuentre también la expresión "goot", será necesario utilizar repeticiones, las cuales se explican más adelante.

Aunque el punto es muy útil para encontrar caracteres que no conocemos, es necesario recordar que corresponde a cualquier carácter y que muchas veces esto no es lo que se requiere. Es muy diferente buscar cualquier carácter que buscar cualquier carácter [alfanumérico](https://es.wikipedia.org/wiki/Alfanum%C3%A9rico) o cualquier dígito o cualquier no-dígito o cualquier no-alfanumérico. Se debe tomar esto en cuenta antes de utilizar el punto y obtener resultados no deseados.

### El signo de admiración "!"

Se utiliza para realizar una "búsqueda anticipada negativa". La construcción de la expresión regular es con el par de paréntesis, el paréntesis de apertura seguida de un signo de interrogación y un signo de exclamación. Dentro de la búsqueda tenemos la expresión regular. Por ejemplo, para excluir exactamente una palabra, habrá que utilizar "^(palabra.+|(?!palabra).\*)$"

### La barra inversa o contrabarra "\"

La [barra inversa](https://es.wikipedia.org/wiki/Barra_inversa) se utiliza para [escapar](https://es.wikipedia.org/w/index.php?title=Secuencias_de_escape_en_C&action=edit&redlink=1) el siguiente carácter de la expresión de búsqueda de forma que este adquiera un significado especial o deje de tenerlo. O sea, la barra inversa no se utiliza nunca por sí sola, sino en combinación con otros caracteres. Al utilizarlo por ejemplo en combinación con el punto "\." este deja de tener su significado normal y se comporta como un carácter literal.

De la misma forma, cuando se coloca la barra inversa seguida de cualquiera de los caracteres especiales que discutiremos a continuación, estos dejan de tener su significado especial y se convierten en caracteres de búsqueda literal.

Como ya se mencionó con anterioridad, la barra inversa también puede darle significado especial a caracteres que no lo tienen. A continuación hay una lista de algunas de estas combinaciones:

* \t — Representa un tabulador.
* \r — Representa el "retorno de carro" o "regreso al inicio" o sea el lugar en que la línea vuelve a iniciar.
* \n — Representa la "nueva línea" el carácter por medio del cual una línea da inicio. Es necesario recordar que en [Windows](https://es.wikipedia.org/wiki/Windows) es necesaria una combinación de \r\n para comenzar una nueva línea, mientras que en [Unix](https://es.wikipedia.org/wiki/Unix) solamente se usa \n y en [Mac\_OS](https://es.wikipedia.org/wiki/Mac_OS) clásico se usa solamente \r.
* \a — Representa una "campana" o "beep" que se produce al imprimir este carácter.
* \e — Representa la tecla "Esc" o "Escape"
* \f — Representa un salto de página
* \v — Representa un tabulador vertical
* \x — Se utiliza para representar caracteres [ASCII](https://es.wikipedia.org/wiki/ASCII) o ANSI si conoce su código. De esta forma, si se busca el símbolo de derechos de autor y la fuente en la que se busca utiliza el conjunto de caracteres [latín-1](https://es.wikipedia.org/wiki/ISO/IEC_8859-1) es posible encontrarlo utilizando \xA9".
* \u — Se utiliza para representar caracteres [Unicode](https://es.wikipedia.org/wiki/Unicode) si se conoce su código. "\u00A2" representa el símbolo de centavos. No todos los motores de Expresiones Regulares soportan Unicode. El .Net Framework lo hace, pero el EditPad Pro no, por ejemplo.
* \d — Representa un dígito del 0 al 9.
* \w — Representa cualquier carácter [alfanumérico](https://es.wikipedia.org/wiki/Alfanum%C3%A9rico).
* \s — Representa un espacio en blanco.
* \D — Representa cualquier carácter que no sea un dígito del 0 al 9.
* \W — Representa cualquier carácter no alfanumérico.
* \S — Representa cualquier carácter que no sea un espacio en blanco.
* \A — Representa el inicio de la cadena. No un carácter sino una posición.
* \Z — Representa el final de la cadena. No un carácter sino una posición.
* \b — Marca la posición de una palabra limitada por espacios en blanco, puntuación o el inicio/final de una cadena.
* \B — Marca la posición entre dos caracteres alfanuméricos o dos no-alfanuméricos.

Nota:

* Algunos lenguajes, como Java, asignan su propio significado a la barra invertida, por lo que deberá repetirse para que sea considerada una expresión regular (ejemplo String expresion="\\d.\\d" para indicar el patrón \d.\d).

### Los corchetes "[ ]"

La función de los corchetes en el lenguaje de las expresiones regulares es representar "clases de caracteres", o sea, agrupar caracteres en grupos o clases. Son útiles cuando es necesario buscar uno de un grupo de caracteres. Dentro de los corchetes es posible utilizar el guion "-" para especificar rangos de caracteres. Adicionalmente, los [metacaracteres](https://es.wikipedia.org/wiki/Metacaracter) pierden su significado y se convierten en literales cuando se encuentran dentro de los corchetes. Por ejemplo, como vimos en la entrega anterior "\d" nos es útil para buscar cualquier carácter que represente un dígito. Sin embargo esta denominación no incluye el punto "." que divide la parte decimal de un número. Para buscar cualquier carácter que representa un dígito o un punto podemos utilizar la expresión regular "[\d.]". Como se hizo notar anteriormente, dentro de los corchetes, el punto representa un carácter literal y no un metacarácter, por lo que no es necesario antecederlo con la barra inversa. El único carácter que es necesario anteceder con la barra inversa dentro de los corchetes es la propia barra inversa. La expresión regular "[\dA-Fa-f]" nos permite encontrar dígitos hexadecimales. Los corchetes nos permiten también encontrar palabras aun si están escritas de forma errónea, por ejemplo, la expresión regular "expresi[oó]n" permite encontrar en un texto la palabra "expresión" aunque se haya escrito con o sin tilde. Es necesario aclarar que sin importar cuantos caracteres se introduzcan dentro del grupo por medio de los corchetes, el grupo sólo le dice al motor de búsqueda que encuentre un solo carácter a la vez, es decir, que "expresi[oó]n" encontrará "expresion" o "expresión".

### La barra "|"

Sirve para indicar una de varias opciones. Por ejemplo, la expresión regular "a|e" encontrará cualquier "a" o "e" dentro del texto. La expresión regular "este|oeste|norte|sur" permitirá encontrar cualquiera de los nombres de los puntos cardinales. La barra se utiliza comúnmente en conjunto con otros caracteres especiales.

### El signo de dólar "$"

Representa el final de la cadena de caracteres o el final de la línea, si se utiliza el modo multi-línea. No representa un carácter en especial sino una posición. Si se utiliza la expresión regular "\.$" el motor encontrará todos los lugares donde un punto finalice la línea, lo que es útil para avanzar entre párrafos.

### El acento circunflejo "^"

Este carácter tiene una doble funcionalidad, que difiere cuando se utiliza individualmente y cuando se utiliza en conjunto con otros caracteres especiales. En primer lugar su funcionalidad como carácter individual: el carácter "^" representa el inicio de la cadena (de la misma forma que el signo de dólar "$" representa el final de la cadena). Por tanto, si se utiliza la expresión regular "^[a-z]" el motor encontrará todos los párrafos que den inicio con una letra minúscula. Cuando se utiliza en conjunto con los corchetes de la siguiente forma "[^\w ]" permite encontrar cualquier carácter que NO se encuentre dentro del grupo indicado. La expresión indicada permite encontrar, por ejemplo, cualquier carácter que no sea alfanumérico o un espacio, es decir, busca todos los símbolos de puntuación y demás caracteres especiales.

La utilización en conjunto de los caracteres especiales "^" y "$" permite realizar validaciones en forma sencilla. Por ejemplo "^\d$" permite asegurar que la cadena a verificar representa un único dígito "^\d\d/\d\d/\d\d\d\d$" permite validar una fecha en formato corto, aunque no permite verificar si es una fecha válida, ya que 99/99/9999 también sería válido en este formato; la validación completa de una fecha también es posible mediante expresiones regulares, como se ejemplifica más adelante.

### Los paréntesis "()"

De forma similar que los corchetes, los paréntesis sirven para agrupar caracteres, sin embargo existen varias diferencias fundamentales entre los grupos establecidos por medio de corchetes y los grupos establecidos por paréntesis:

* Los caracteres especiales conservan su significado dentro de los paréntesis.
* Los grupos establecidos con paréntesis establecen una "etiqueta" o "punto de referencia" para el motor de búsqueda que puede ser utilizada posteriormente como se denota más adelante.
* Utilizados en conjunto con la barra "|" permite hacer búsquedas opcionales. Por ejemplo la expresión regular "al (este|oeste|norte|sur) de" permite buscar textos que den indicaciones por medio de puntos cardinales, mientras que la expresión regular "este|oeste|norte|sur" encontraría "este" en la palabra "esteban", no pudiendo cumplir con este propósito.
* Utilizados en conjunto con otros caracteres especiales que se detallan posteriormente, ofrece funcionalidad adicional.

### El signo de interrogación "?"

El signo de interrogación tiene varias funciones dentro del lenguaje de las expresiones regulares. La primera de ellas es especificar que una parte de la búsqueda es opcional. Por ejemplo, la expresión regular "ob?scuridad" permite encontrar tanto "oscuridad" como "obscuridad". En conjunto con los paréntesis redondos permite especificar que un conjunto mayor de caracteres es opcional; por ejemplo "Nov(\.|iembre|ember)?" permite encontrar tanto "Nov" como "Nov.", "Noviembre" y "November". Como se mencionó anteriormente, los paréntesis nos permiten establecer un "punto de referencia" para el motor de búsqueda. Sin embargo, algunas veces, no se desea utilizarlos con este propósito, como en el ejemplo anterior "Nov(\.|iembre|ember)?". En este caso el establecimiento de este punto de referencia (que se detalla más adelante) representa una inversión inútil de recursos por parte del motor de búsqueda. Para evitarlo se puede utilizar el signo de pregunta de la siguiente forma: "Nov(?:\.|iembre|ember)?". Aunque el resultado obtenido será el mismo, el motor de búsqueda no realizará una inversión inútil de recursos en este grupo, sino que lo ignorará. Cuando no sea necesario reutilizar el grupo, es aconsejable utilizar este formato. De forma similar, es posible utilizar el signo de pregunta con otro significado: Los paréntesis definen grupos "anónimos", sin embargo el signo de pregunta en conjunto con los paréntesis triangulares "<>" permite "nombrar" estos grupos de la siguiente forma: "^(?<Día>\d\d)\/(?<Mes>\d\d)\/(?<Año>\d\d\d\d)$"; Con lo cual se le especifica al motor de búsqueda que los primeros dos dígitos encontrados llevarán la etiqueta "Día", los segundos la etiqueta "Mes" y los últimos cuatro dígitos llevarán la etiqueta "Año".

NOTA: a pesar de la complejidad y flexibilidad dada por los caracteres especiales estudiados hasta ahora, en su mayoría nos permiten encontrar solamente un carácter a la vez, o un grupo de caracteres a la vez. Los metacaracteres enumerados en adelante permiten establecer repeticiones.

### Las llaves "{}"

Comúnmente las llaves son caracteres literales cuando se utilizan por separado en una expresión regular. Para que adquieran su función de metacaracteres es necesario que encierren uno o varios números separados por coma y que estén colocados a la derecha de otra expresión regular de la siguiente forma: "\d{2}" Esta expresión le dice al motor de búsqueda que encuentre dos dígitos contiguos. Utilizando esta fórmula podríamos convertir el ejemplo "^\d\d/\d\d/\d\d\d\d$" que servía para validar un formato de fecha en "^\d{2}/\d{2}/\d{4}$" para una mayor claridad en la lectura de la expresión.

"\d{2,4}" Esta forma añade un segundo número separado por una coma, el cual indica al motor de búsqueda que como máximo debe aparecer 4 veces la expresión regular \d. Los posibles valores son:

* "^\d\d$" (mínimo 2 repeticiones)
* "^\d\d\d$"(tiene 3 repeticiones, por lo tanto entra en el rango 2-4)
* "^\d\d\d\d$" (máximo 4 repeticiones)

Nota: aunque esta forma de encontrar elementos repetidos es muy útil, algunas veces no se conoce con claridad cuantas veces se repite lo que se busca o su grado de repetición es variable. En estos casos los siguientes metacaracteres son útiles.

### El asterisco "\*"

El asterisco sirve para encontrar algo que se encuentra repetido 0 o más veces. Por ejemplo, utilizando la expresión "[a-zA-Z]\d\*" será posible encontrar tanto "H" como "H1", "H01", "H100" y "H1000", es decir, una letra seguida de un número indefinido de dígitos. Es necesario tener cuidado con el comportamiento del asterisco, ya que este, por defecto, trata de encontrar la mayor cantidad posible de caracteres que correspondan con el patrón que se busca. De esta forma si se utiliza "\(.\*\)" para encontrar cualquier cadena que se encuentre entre paréntesis y se lo aplica sobre el texto "Ver (Fig. 1) y (Fig. 2)" se esperaría que el motor de búsqueda encuentre los textos "(Fig. 1)" y "(Fig. 2)", sin embargo, debido a esta característica, en su lugar encontrará el texto "(Fig. 1) y (Fig. 2)". Para obtener el resultado deseado se debe utilizar el asterisco en conjunto con el signo de interrogación de la siguiente forma: "\(.\*?\)" Esto es equivalente a decirle al motor de búsqueda que "Encuentre un paréntesis de apertura y luego encuentre cualquier secuencia de caracteres hasta que encuentre un paréntesis de cierre".

### El signo de suma "+"

Se utiliza para encontrar una cadena que se encuentre repetida una o más veces. A diferencia del asterisco, la expresión "[a-zA-Z]\d+" encontrará "H1" pero no encontrará "H". También es posible utilizar este metacarácter en conjunto con el signo de interrogación para limitar hasta donde se efectúa la repetición.

# Operadores de expresiones regulares

**Existen unas normas para el uso de los operadores de expresiones regulares:**

* En las consultas, los asteriscos (\*), puntos (.) y signos de interrogación (?) se comportan igual que en Windows. El asterisco (\*) representa cualquier número de caracteres. El punto (.) representa el final de una cadena y el signo de interrogación (?), cualquier carácter individual.
* Cualquier carácter, excepto el asterisco (\*), el punto (.), el signo de interrogación (?) y la barra vertical (|), coincide de manera predeterminada consigo mismo.
* Las expresiones regulares pueden incluirse entre comillas (" ") de apertura y cierre y deben incluirse entre comillas si contienen un espacio o paréntesis de cierre ( ) ).

Si van precedidos de una barra vertical (|), los siguientes caracteres adquieren un significado especial:

|  |  |
| --- | --- |
| **Carácter** | **Descripción** |
| ( | Abre un grupo. Debe ir seguido del paréntesis de cierre correspondiente ( ) ). |
| ) | Cierra un grupo. Debe ir precedido del paréntesis de apertura correspondiente ( ( ). |
| [ | Abre una clase de carácter. Debe ir seguido del corchete de cierre correspondiente (]). |
| ] | Cierra una clase de carácter. Debe ir precedido del corchete de apertura correspondiente ([). |
| { | Abre una correspondencia exacta. Debe ir seguido de la llave de cierre correspondiente (}). |
| } | Cierra una correspondencia exacta. Debe ir precedido de la llave de apertura correspondiente ({). |
| , | Separa cláusulas OR. |
| \* | Corresponde a cero o más ocurrencias de la expresión precedente. |
| ? | Corresponde a cero o una ocurrencia de la expresión precedente. |
| + | Corresponde a una o más ocurrencias de la expresión precedente. |
| Todos los demás | Coinciden consigo mismos. |

**Corchetes**

Si se utilizan entre corchetes ([ ]), los siguientes caracteres adquieren un significado especial:

|  |  |
| --- | --- |
| **Carácter** | **Descripción** |
| ^ | Debe ser el primer carácter. Coincide con todo excepto con las clases siguientes. |
| ] | Cierra la clase. Sólo puede ir precedido de un símbolo de intercalación (^). |
| - | Operador de rango. Precedido y seguido por otros caracteres. |
| Todos los demás | Coinciden consigo mismos, inician o acaban un rango. |

**Llaves**

Si se escriben entre llaves ({ }), los siguientes caracteres adquieren los significados que se indican a continuación.

|  |  |
| --- | --- |
| **Carácter** | **Descripción** |
| m | Coincide exactamente con *m* ocurrencias de la expresión precedente (0< m < 256). |
| m, | Coincide como mínimo con *m* ocurrencias de la expresión precedente (1< m < 256). |
| m,n | Coincide con entre *m* y *n* ocurrencias de la expresión precedente, ambas inclusive (0 < m < 256, 0 < n < 256). |

Para hacer que el asterisco (\*), el punto (.) y el signo de interrogación (?) coincidan consigo mismos, escríbalos entre corchetes. Por ejemplo, para buscar "hola?", escriba **hola[?]** en la consulta.

# Ejemplos de expresiones regulares

Puede escribir expresiones regulares en cuadros de texto para valores de campos a fin de extraer campos de eventos de registro.

Las expresiones que escriba deben usar la sintaxis de expresiones regulares de Java.

| **Operadores de caracteres** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| \ | Es el carácter de escape para los caracteres especiales. |
| \b | Límite de palabra |
| \B | No es límite de palabra |
| \d | Un dígito |
| \D | Un carácter que no es dígito |
| \n | Nueva línea |
| \r | Carácter de retorno |
| \s | Un espacio |
| \S | Cualquier carácter, excepto espacio en blanco |
| \t | Tabulador |
| \w | Un carácter alfanumérico o guion bajo |
| \W | Un carácter no alfanumérico o guion bajo |
|  |  |

Por ejemplo, si tiene la cadena 1234-5678 y aplica las siguientes expresiones regulares, obtendrá:

| **Expresión regular** | **Resultado** |
| --- | --- |
| \d | 1 |
| \d+ | 1234 |
| \w+ | 1234 |
| \S | 1234-5678 |

| **Operadores cuantificadores** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| . | Cualquier carácter, excepto una línea nueva |
| \* | Cero o más con la máxima extensión posible |
| ? | Cero o un carácter OR lo más corto posible |
| + | Uno o más |
| {<n>} | Exactamente <n> veces |
| {<n>,<m>} | <n> a <m> veces |

Por ejemplo, si tiene la cadena aaaaa y aplica las siguientes expresiones regulares

| **Expresión regular** | **Resultado** |
| --- | --- |
| . | a |
| \* | aaaaa |
| .\*? | aaaaa |
| .{1} | a |
| .{1,2} | aa |

| **Operadores de combinación** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| .\* | Cualquier cosa |
| .\*? | Cualquier cosa lo más breve posible antes de |

Por ejemplo, si tiene la cadena a b 3 hi d hi y aplica las siguientes expresiones regulares

| **Expresión regular** | **Resultado** |
| --- | --- |
| a.\* hi | b 3 hi d |
| a .\*? hi | b 3 |

| **Operadores lógicos** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| ^ | Comienzo de una línea OR no si está entre corchetes |
| $ | Fin de una línea |
| () | Encapsulación |
| [] | Un carácter entre corchetes |
| | | OR |
| - | Intervalo |
| \A | Comienzo de una cadena |
| \Z | Fin de una cadena |

Por ejemplo, si aplica las siguientes expresiones regulares

| **Expresión regular** | **Resultado** |
| --- | --- |
| (hola)? | Contiene hola OR no contiene hola |
| (a|b|c) | a OR b OR c |
| [a-cp] | a OR b OR c OR p |
| palabra $ | Finaliza con palabra seguido por nada más |

| **Operadores de lectura previa** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| ?= | Lectura previa positiva (incluye) |
| ?!= | Lectura previa negativa (no incluye) |

Por ejemplo, si aplica las siguientes expresiones regulares

| **Expresión regular** | **Resultado** |
| --- | --- |
| is (?=\w+)\w{2} primario | es FT primario? falso |
| opid=(?!WFU-1fecf8f9)\S+ | WFU-3c9bb994 |

| **Otros ejemplos de expresiones regulares** | |
| --- | --- |
| **Expresión regular** | **Descripción** |
| [xyz] | x, y, o z |
| (información|advertencia|error) | información, advertencia o error |
| [a-z] | Una letra minúscula |
| [^a-z] | No una letra minúscula |
| [a-z]+ | Una o más letras minúsculas |
| [a-z]\* | Cero o más letras minúsculas |
| [a-z]? | Cero o una letra minúscula |
| [a-z] {3} | Exactamente tres letras minúsculas |
| [\d] | Un dígito |
| \d+$ | Uno o más dígitos seguidos por fin del mensaje |
| [0-5] | Un número de 0 a 5 |
| \w | Un carácter de palabra (letra, dígito o guion bajo) |
| \s | Espacio en blanco |
| \S | Cualquier carácter, excepto espacio en blanco |
| [a-zA-Z0-9]+ | Uno o más caracteres alfanuméricos |
| ([a-z] {2,} [0-9] {3,5}) | Dos o más letras seguidas por tres a cinco números |