

# [EXPRESSÕES REGULARES]

UMA ABORDAGEM DIVERTIDA

5ª Edição – Revisada e Ampliada

**Aurelio Marinho Jargas**

Copyright © 2006, 2008, 2009, 2012, 2016 da Novatec Editora Ltda.

Todos os direitos reservados e protegidos pela Lei 9610 de 19/02/1998.  
É proibida a reprodução desta obra, mesmo parcial, por qualquer processo,  
sem prévia autorização, por escrito, do autor e da Editora.

Editor: Rubens Prates

Assistente editorial: Priscila A. Yoshimatsu

Revisão de textos: Marta Almeida de Sá

Capa, projeto gráfico e editoração eletrônica: Karine Hermes

Adaptação do projeto gráfico: Carolina Kuwabata

Ilustração da capa: Vinicius Vogel

ISBN: 978-85-7522-474-8

Histórico de impressões:

|               |   |
|---------------|---|
| Janeiro/2016  | Quinta edição (ISBN: 978-85-7522-474-8)   |
| Junho/2014    | Primeira reimpressão                      |
| Outubro/2012  | Quarta edição (ISBN: 978-85-7522-337-6)   |
| Junho/2011    | Primeira reimpressão                      |
| Setembro/2009 | Terceira edição (ISBN: 978-85-7522-212-6) |
| Julho/2008    | Segunda edição (ISBN: 978-85-7522-173-0)  |
| Novembro/2006 | Primeira edição (ISBN: 85-7522-100-0)     |

Novatec Editora Ltda.

Rua Luís Antônio dos Santos 110

02460-000 São Paulo, SP – Brasil

Tel.: +55 11 2959-6529

E-mail: [novatec@novatec.com.br](mailto:novatec@novatec.com.br)

Site: [www.novatec.com.br](http://www.novatec.com.br)

Twitter: [twitter.com/novateceditora](https://twitter.com/novateceditora)

Facebook: [facebook.com/novatec](https://facebook.com/novatec)

LinkedIn: [linkedin.com/in/novatec](https://linkedin.com/in/novatec)

# Capítulo 1

## Introdução



Olá. Que tal esquecer um pouco a rotina e a realidade e fazer uma viagem ao interior de sua mente? Descobrir conceitos novos, diferentes. Ao voltar, as coisas não serão mais tão normais quanto antes, pois símbolos estranhos farão parte de seu dia a dia.

Inspirado pelo funcionamento de seus próprios neurônios, descubra o fascinante mundo abstrato das expressões regulares.

### Objetivo

Neste nosso mundo tecnoinformatizado, onde o acesso rápido à informação desejada é algo crucial, temos nas expressões regulares uma mão amiga, que quanto mais refinada for sua construção, mais preciso e rápido será o resultado, diferenciando aqueles que as dominam daqueles que perdem horas procurando por dados que estão ao alcance da mão.

O assunto é algo bem peculiar, pois apesar de a maioria das linguagens de programação, programas e editores de texto mais utilizados possuírem esse recurso, poucos o dominam, principalmente pelo fato de a documentação sobre o assunto, quando existente, ser enigmática e pouco didática, ou simplesmente se resumir a listagens, sem explicar os conceitos. Esta obra nasceu dessa necessidade e tem como objetivo preencher essa lacuna, sendo uma documentação completa e didática para iniciantes, tipo tutorial, e um guia de referência para os já iniciados.

Este livro é a primeira publicação em português totalmente dedicada ao assunto, e espero que esse pioneirismo traga muitos frutos, inclusive outras publicações sobre o tema, para difundir e desmistificar o uso das expressões regulares.

## Sobre o livro

A primeira parte é o “feijão com arroz”, indicada àqueles que desconhecem ou ainda não se sentem à vontade para criar suas próprias expressões regulares. Faremos um *tour* por todas as pecinhas que compõem esse mundo fantástico, explicando didaticamente, do zero, o que são, de onde vieram, para que servem e como utilizá-las (Exemplos! Exemplos!).

Após ler e entender essa primeira parte, algo como

```
^ *[A-Za-z0-9_]+:(.*)$
```

vai fazer parte de sua realidade, sem lhe causar pânico.

A segunda parte é a “feijoada”, para aqueles que querem uma experiência mais intensa. Mergulharemos de cabeça e entenderemos de vez essa maquininha esquisita. São as explicações dos conceitos envolvidos, bem como táticas e dicas para você realmente entender e otimizar seu uso das expressões regulares. Ao final da leitura, você entenderá por que

```
^[^:]+:([A-Za-z]+):
```

é melhor que

```
.*:(.*):
```

Mas note que, tudo isso, sem viajar muito nos detalhes intrínsecos e sem conhecer os becos escuros que você talvez nunca precisará saber que existem. Acima de tudo este é um livro prático. É para ler e fazer suas expressões. Isso não o torna superficial, apenas direto.

Com tudo isso, temos diversas tabelas e listagens que servem para ser consultadas rapidamente em caso de dúvida ou esquecimento. Relaxe, não é um bicho de [0-9]+ cabeças... Vamos bater um papo descontraído sobre o assunto. Então respire fundo, desligue a TV, olhe fixamente para estas letras e vamos!

## Apresentando as Expressões Regulares

Então, para podermos começar nossa viagem, nada como uma apresentação de nosso assunto, pois, afinal de contas, que raios são estas expressões?

Bem resumido, uma expressão regular é um método formal de se especificar um padrão de texto.

Mais detalhadamente, é uma composição de símbolos, caracteres com funções especiais, que, agrupados entre si e com caracteres literais, formam uma sequência, uma expressão. Essa expressão é interpretada como uma regra que indicará sucesso se uma entrada de dados qualquer casar com essa regra, ou seja, obedecer exatamente a todas as suas condições.

Ou como variações aceitas também pode-se afirmar que é:

- uma maneira de procurar um texto que você não lembra exatamente como é, mas tem ideia das variações possíveis;
- uma maneira de procurar um trecho em posições específicas como no começo ou no fim de uma linha, ou palavra;
- uma maneira de um programador especificar padrões complexos que podem ser procurados e casados em uma cadeia de caracteres;
- uma construção que utiliza pequenas ferramentas, feita para obter determinada sequência de caracteres de um texto.

Ou ainda, didaticamente falando, é:

- Como o brinquedo LEGO, várias pecinhas diferentes, cada uma com sua característica, que juntas compõem estruturas completas e podem ser arranjadas com infinitas combinações diferentes.

- Como um jogo de truco, com as cartas normais e as quentes: gato, copas, espadilha e mole, que são especiais e têm uma ordem de grandeza.
- Como um quebra-cabeça, sempre tem solução, às vezes óbvia, às vezes difícil, mas, decifrando as partes, junta-se tudo e chega-se ao todo.
- Como um jogo, no começo é difícil, mas após conhecer todas as regras, basta jogar e curtir.
- Como uma receita culinária, com seus ingredientes e uma ordem correta para adicioná-los à mistura.
- Como consertar carros. Você tem várias peças e várias ferramentas. Dependendo do tipo de peça, há uma ferramenta certa para você lidar com ela. E dependendo da sua localização, você tem de incrementar a ferramenta com mais barras e cotovelos para alcançá-la.
- Como o alfabeto. Você aprende primeiro as letras individualmente. Depois as sílabas, as palavras, as frases e finalmente os textos. Mas no fundo são apenas letras.

Acima de tudo, assim como um sorvete no domingo ensolarado, uma expressão regular é:

Divertida!

**Divertida? Tá louco?**  
**Todos aqueles símbolos**  
**estranhos...**



Calma... É normal estranharmos ou até repudiarmos aquilo que ainda não conhecemos ou não dominamos bem. Como diria o vovô Simpson no meio da multidão: “Vamos destruir aquilo que não entendemos!”

Ao final da leitura, ficará claro que as expressões são apenas pequenos pedacinhos simples que agrupados formam algo maior. O importante é você compreender bem cada um individualmente, e depois apenas lê-los em sequência. Lembre-se do alfabeto: são apenas letras...

## História



Sim! Vou te contar uma história. A fecundação dessas expressões aconteceu no ano de 1943, quando os “pais”, dois neurologistas, publicaram um estudo que teorizava o funcionamento dos nossos neurônios. Sentiu o drama? Nosso assunto é nobre desde a sua origem.

Anos depois, o “parteiro”, um matemático, descreveu algebricamente os modelos desse estudo utilizando símbolos para representar seus recém-criados grupos regulares (do inglês “regular sets”). Com a criação dessa notação simbólica, nasceram as expressões regulares, que durante toda a sua infância e juventude (cerca de vinte anos) foram muito estudadas pelos matemáticos da época.

Mas o encontro com o computador só aconteceu mesmo em 1968, em um algoritmo de busca utilizado no editor de textos `qed`, que depois virou o `ed`, editor padrão dos primeiros sistemas Unix. Este `ed` tinha o comando de contexto `g`, que aceitava expressões regulares e um comando `p`. Sua sintaxe ficava `g/RE/p` (“Global Regular Expression Print”), que deu origem ao aplicativo `grep`, que por sua vez originou o `egrep`.

Outros filhos como o `sed` e o `awk` também apareceram, cada um implementando as expressões do seu próprio jeito; e finalmente em 1986 foi criado o divisor de águas, um pacote pioneiro em C chamado `regex` que tratava das expressões regulares e qualquer um poderia incluí-lo em seu próprio programa, de graça. Opa! Falaram as palavras mágicas: de graça. Aí não teve mais volta, as expressões caíram no gosto popular e cada vez mais e mais programas e linguagens as utilizam.

**Curiosidade:** apesar de esse assunto ser antigo, o que vamos ver aqui basicamente é o mesmo que um estudante veria 25 anos atrás. É um conceito consistente, que não sofre alterações com o passar do tempo.

## Terminologia

E se eu te disser que “ERs são metacaracteres que casam um padrão”? Não entendeu?

Bem, como expressões regulares é um termo muito extenso, daqui em diante, chamarei apenas de ER (ê-érre) para simplificar a leitura. Outras nomenclaturas que podem ser encontradas em outras fontes são expreg, regexp, regex e RE. Particularmente, regex é uma boa escolha para usar em ferramentas de busca na Internet.

E como estamos falando de termos, tem mais alguns novos que farão parte de nossa conversa. Lembra que as expressões são formadas por símbolos e caracteres literais? Esses símbolos são chamados de metacaracteres, pois possuem funções especiais, que veremos detalhadamente adiante.

Outro termo que é interessante e às vezes pode assustar um leitor meio distraído é o casar (“match”). Casamento aqui não é juntar os trapos, mas, sim, o ato de bater, conferir, combinar, igualar, encontrar, encaixar, equiparar. É como em um caixa 24 horas, em que você só retirará o dinheiro se sua senha digitada casar com aquela já cadastrada no banco.

Também temos o padrão (“pattern”), que é nosso objetivo quando fazemos uma ER: casar um padrão. Esse padrão pode ser uma palavra, várias, uma linha vazia, um número, ou seja, o que quer que precise ser encontrado pela nossa ER.

E ainda tem o robozinho, que é uma referência ao compilador e interpretador das expressões regulares, o código que vai ler, checar, entender e aplicar sua ER no texto desejado. Como exemplo, para programas em C o robozinho é a biblioteca regex, que faz todo o serviço.



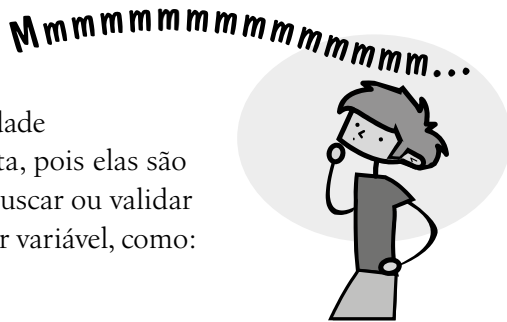
## Para que servem?

Basicamente servem para você dizer algo abrangente de forma específica. Definido seu padrão de busca, você tem uma lista (finita ou não) de possibilidades de casamento. Em um exemplo rápido, `[rgp]ato` pode casar rato, gato e pato. Ou seja, sua lista “abrange especificamente” essas três palavras, nada mais.

Na prática, as expressões regulares servem para uma infinidade de tarefas, é difícil fazer uma lista, pois elas são úteis sempre que você precisar buscar ou validar um padrão de texto que pode ser variável, como:

- data
- horário
- número IP
- nome de pessoa
- endereço de e-mail
- endereço de Internet
- nome de usuário e senha
- declaração de uma função()
- dados na coluna N de um texto
- dados que estão entre `<tags></tags>`
- campos específicos de um texto tabulado
- número de telefone, RG, CPF, cartão de crédito
- dados que estão apenas no começo ou no fim da linha

E mais uma infinidade de outros padrões que não podem ser especificados com caracteres literais.



Um exemplo prático: você tem uma lista diária de acesso de usuários que entraram em seu sistema, onde consta, em cada linha, o horário do acesso e o login do usuário, algo como:

```
05:15 ernesto
08:39 ricardo
10:32 patricia
14:59 gabriel
16:27 carla
22:23 marcelo
```

Como fazer para buscar automaticamente apenas os usuários que acessaram o sistema no período da tarde (meio-dia às seis)? Você tem várias opções, desde procurar uma a uma manualmente até fazer um programa que compare os primeiros caracteres de cada linha, mas, falando de algo prático e rápido, que não exija conhecimentos de programação, a ER é simplesmente `^[2-8]`.

O que significa essa  
sopa de letrinhas?



Caaaaaaalma. Acompanhe o próximo tópico e vamos conhecer todos os metacaracteres, essas coisinhas úteis que facilitam nossa vida.



# Capítulo 2

## Os metacaracteres

Então, para já matar sua curiosidade, aqui estão os tão falados metacaracteres-padrão que serão nossos personagens das próximas páginas:

. ? \* + ^ \$ | [ ] { } ( ) \

E aí, sentiu um frio na barriga? Cada simbolozinho desses tem sua função específica, que pode mudar dependendo do contexto no qual está inserido, e podemos agregá-los uns com os outros, combinando suas funções e fazendo construções mais complexas. Olha, ainda dá tempo de fechar o livro e voltar a assistir à novela...

Então deixe eu te assustar mais um pouquinho. Além destes, temos outros metacaracteres estendidos que foram criados posteriormente, pois tarefas mais complexas requisitavam funções mais específicas ainda. E para terminar de complicar, sua sintaxe de utilização não é a mesma para todos os programas que suportam expressões regulares.

Bem, já que você não desistiu (eu tentei), vamos logo ao que interessa, e para começar vamos dar nomes aos bois. Leia, releia e treleia esta lista, repetindo para si mesmo e associando o nome ao símbolo, pois estas palavras farão parte de sua vida, de sua rotina. Acostume-se com os nomes e não os mude.

| Metacaractere | Nome         | Metacaractere | Nome        |
|---------------|--------------|---------------|-------------|
| .             | Ponto        | ^             | Circunflexo |
| [ ]           | Lista        | \$            | Cifrão      |
| [^]           | Lista negada | \b            | Borda       |
| ?             | Opcional     | \             | Escape      |
| *             | Asterisco    |               | Ou          |
| +             | Mais         | ()            | Grupo       |
| { }           | Chaves       | \1            | Retrovisor  |

Agora que sabemos como chamar nossos amigos novos, veremos uma prévia, um apanhado geral de todos os metacaracteres e suas funções. Eles estão divididos em quatro grupos distintos, de acordo com características comuns entre eles.

### Representantes

| Metacaractere | Nome         | Função                         |
|---------------|--------------|--------------------------------|
| .             | Ponto        | Um caractere qualquer          |
| [...]         | Lista        | Lista de caracteres permitidos |
| [^...]        | Lista negada | Lista de caracteres proibidos  |

### Quantificadores

| Metacaractere  | Nome      | Função                   |
|----------------|-----------|--------------------------|
| ?              | Opcional  | Zero ou um               |
| *              | Asterisco | Zero, um ou mais         |
| +              | Mais      | Um ou mais               |
| { <i>n,m</i> } | Chaves    | De <i>n</i> até <i>m</i> |

## Âncoras

| Metacaractere | Nome        | Função                   |
|---------------|-------------|--------------------------|
| ^             | Circunflexo | Início da linha          |
| \$            | Cifrão      | Fim da linha             |
| \b            | Borda       | Início ou fim de palavra |

## Outros

| Metacaractere | Nome       | Função                        |
|---------------|------------|-------------------------------|
| \c            | Escape     | Torna literal o caractere c   |
|               | Ou         | Ou um ou outro                |
| (...)         | Grupo      | Delimita um grupo             |
| \1...\9       | Retrovisor | Texto casado nos grupos 1...9 |



**Aaaaah, então  
? e \* eu já uso na  
linha de comando!**

Opa, não confunda! Os curingas usados na linha de comando para especificar nomes de arquivos, como \*.txt, relatorio.{txt,doc} e foto-?.jpg não são expressões regulares. São curingas específicos de nomes de arquivo, e, apesar de parecidos, são outra coisa e os significados de seus

símbolos são diferentes dos das expressões. Então o melhor que você faz agora é esquecer esses curingas, senão eles podem confundir-lo e atrapalhar seu aprendizado.

Ah, antes que eu me esqueça: para testar os exemplos que veremos a seguir, acesse o site do livro: [www.piazinho.com.br](http://www.piazinho.com.br). Há uma ferramenta especial para você testar todos os exemplos, além de poder fazer suas próprias expressões. Experimente!

## Metacaracteres tipo Representante

O primeiro grupo de metacaracteres que veremos são os do tipo representante, ou seja, são metacaracteres cuja função é representar um ou mais caracteres.

Também podem ser encarados como apelidos, links ou qualquer outra coisa que lhe lembre essa associação entre elementos. Todos os metacaracteres desse tipo casam a posição de um único caractere, e não mais que um.

### Ponto: o necessitado .

O ponto é nosso curinga solitário, que está sempre à procura de um casamento, não importa com quem seja. Pode ser um número, uma letra, um Tab, um @, o que vier ele traça, pois o ponto casa qualquer coisa.

Suponhamos uma ER que contenha os caracteres “fala” e o metacaractere ponto, assim: “fala.”. No texto a seguir, essa ER casaria tudo o que está sublinhado:

“Olha, com vocês me pressionando, a fala não vai sair natural. Eu não consigo me concentrar na minha fala. Aliás, isso é um falatório, pois nunca vi um comercial com tantas falas assim. Vou me queixar com o problemasnafala@medicos.com.br.”

Nesse pequeno trecho de texto, nossa ER casou cinco vezes, tendo o ponto casado com os seguintes caracteres: “.ts@”.



Como exemplos de uso do ponto, em um texto normal em português, você pode procurar palavras que você não se lembra se acentuou ou não, que podem começar com maiúsculas ou não ou que foram escritas errado:

| Expressão | Casa com                             |
|-----------|--------------------------------------|
| n.o       | não, nao, ...                        |
| .eclado   | teclado, Teclado, ...                |
| e.tendido | estendido, extendido, eztendido, ... |

Ou, para tarefas mais específicas, procurar horário com qualquer separador ou com marcações (“tags”) HTML:

| Expressão | Casa com                 |
|-----------|--------------------------|
| 12.45     | 12:45, 12 45, 12.45, ... |
| <.>       | <B>, <i>, <p>, ...       |



### RESUMÃO

- O ponto casa com qualquer coisa.
- O ponto casa com o ponto.
- O ponto é um curinga para casar um caractere.

## Lista: a exigente [...]

Bem mais exigente que o ponto, a lista não casa com qualquer um. Ela sabe exatamente o que quer, e não aceita nada diferente daquilo, a lista casa com quem ela conhece.

Ela guarda dentro de si os caracteres permitidos para casar, então algo como [aeiou] limita nosso casamento a letras vogais.

No exemplo anterior do ponto, sobre acentuação, tínhamos a ER n.o. Além dos casamentos desejados, ela é muito abrangente, e também casa coisas indesejáveis como neo, n-o, n5o e n o.

Para que nossa ER fique mais específica, trocamos o ponto pela lista, para casar apenas “nãõ” e “nao”, veja:

n[ãa]o

E, assim como o n.o, todos os outros exemplos anteriores do ponto casam muito mais que o desejado, justo pela sua natureza promíscua.

Por isso que nos  
exemplos tinha os  
três pontinhos  
no final?



Exatamente, eles indicam que havia mais possibilidades de casamento. Como o ponto casa com qualquer coisa, ele é nada específico. Então vamos impor limites às ERs:

| Expressão    | Casa com             |
|--------------|----------------------|
| n[ãa]o       | não, nao             |
| [Tt]eclado   | Teclado, teclado     |
| e[sx]tendido | estendido, extendido |
| 12[:. ]45    | 12:45, 12.45, 12 45  |
| <[BIP]>      | <B>, <I>, <P>        |

Mas e aquele ponto  
na ER da hora,  
não casa com  
qualquer coisa?



Pegadinha! Não. Registre em algum canto de seu cérebro: dentro da lista, todo mundo é normal. Repetindo: dentro da lista, todo mundo é normal. Então aquele ponto é simplesmente um ponto normal, e não um metacaractere.

No exemplo de marcação <[BIP]>, vemos que as ERs são sensíveis a maiúsculas e minúsculas, então, se quisermos mais possibilidades, basta incluí-las:

| Expressão  | Casa com                     |
|------------|------------------------------|
| <[BIPbip]> | <B>, <I>, <P>, <b>, <i>, <p> |



## Intervalos em listas

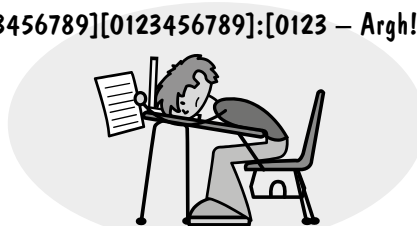
Por enquanto, vimos que a lista abriga todos os caracteres permitidos em uma posição. Como seria uma lista que dissesse que em uma determinada posição poderia haver apenas números?

**Peraí que essa eu sei... deixa ver...  
[0123456789]. Acertei?**



Sim! Então, para casar uma hora, qualquer que ela seja, fica como? Lembre que o formato é hh:mm.

**Tá. [0123456789][0123456789]:[0123 – Argh! QUE SACO!**



Pois é! Assim também pensaram nossos ilustres criadores das ERs, e, para evitar esse tipo de listagem extensa, dentro da lista temos o conceito de intervalo.

Lembra que eu disse para você memorizar que dentro da lista, todo mundo é normal? Pois é, aqui temos a primeira exceção à regra. Todo mundo, fora o traço. Se tivermos um traço (-) entre dois caracteres, isso representa todo o intervalo entre eles.

Não entendeu? É assim, olhe:

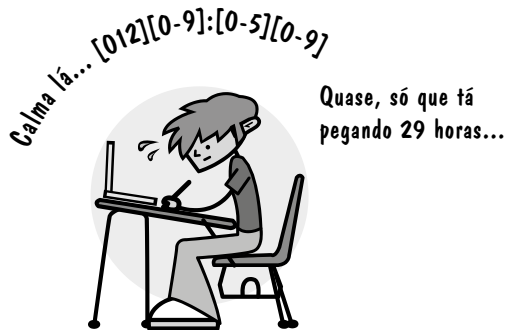
[0123456789] é igual a [0-9]

É simples assim. Aquele tracinho indica um intervalo, então 0-9 se lê: de zero a nove.

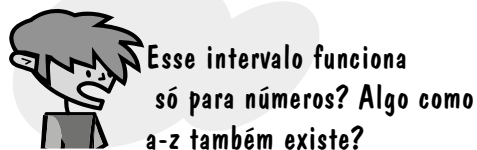
Voltando a nossa ER da hora, poderíamos fazer

```
[0-9][0-9]:[0-9][0-9]
```

mas veja que não é específico o bastante, pois permite uma hora como 99:99, que não existe. Como poderíamos fazer uma ER que casasse no máximo 23 horas e 59 minutos?



Excelente! Com o que aprendemos até agora, esse é o máximo de precisão que conseguimos. Mais adiante, quem poderá nos ajudar será o ou. Depois voltamos a esse problema.



Era isso que eu ia falar agora. Sim, qualquer intervalo é válido, como a-z, A-Z, 5-9, a-f, :-@ etc.

