

# Projeto Integrador Transdisciplinar em Análise e Desenvolvimento de Sistemas I.



**Conceudista:** Prof. Me. Edeilson Silva

**Revisão Textual:** Prof.<sup>a</sup> Dra. Selma Aparecida Cesarin

**Material Teórico**

**Material Complementar**

**DESAFIO**

---

**Situação-Problema 1**

**Situação-Problema 2**

**Situação-Problema 3**

**Problema em Foco**

**ATIVIDADE**

---

**Atividade de Entrega**

## **REFERÊNCIAS**

---

### **Referências**

# Material Teórico

Olá, estudante!

Vamos iniciar a disciplina abordando os conceitos necessários para que você possa realizar a atividade através das situações-problema mais à frente.

**(i) Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo online para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.**

## Introdução

Esta Disciplina tem como objetivo apresentar temas atuais que favoreçam o seu aprendizado e os relate às Disciplinas cursadas por você até o momento. Nesse sentido, trabalharemos com a temática das Expressões Regulares (ER), também conhecidas como *Regular Expressions (RegEx)*. Elas permitem que padrões de caracteres sejam buscados, validados e substituídos, quando necessário.

Segundo Jargas (2012, p. 19) uma expressão regular é um método formal de se especificar um padrão de texto. Fitzgerald (2012) acrescenta a ideia de que essas *strings* (texto) são especialmente codificadas e utilizadas como padrões para combinar com outros conjuntos de *strings*. Na programação, *string* é um conjunto de caracteres cujo valor é texto. Dessa forma, grande parte das linguagens de programação, interfaces de desenvolvimento (IDE) e editores de texto têm esse recurso.

Trabalhar com expressões regulares exige bastante treino, pois haverá momentos que um simples: `\d` (abreviação que corresponde a qualquer dígito de 0 a 9) poderá resolver um problema, enquanto em outros, uma expressão mais complexa como `^\d{1-9}\d{2}\d{1} (\d{8}\d{9}\d{0-9}\d{4})-\d{0-9}\d{4}\$`, será necessária. Essa expressão corresponde ao número de telefone móvel brasileiro com 11 dígitos, incluindo os parênteses para a identificação da cidade, espaço e um hífen entre os números. Veja o exemplo: (99) 99999-9999.

Para tentar deixar as coisas mais claras, vamos pensar em algo simples utilizando ER: imagine um hipotético viajante que está concludo suas férias na cidade de Dublin. Ele está no centro da cidade e precisa chegar ao aeroporto. Sua busca por horários trouxe uma lista com paradas e horários do ônibus 782. Agora basta procurar embarques entre 4h30 e 4h35.

**Tabela 1 – Paradas de ônibus**

<i>Location</i>	<i>Stop/Zone</i>	<i>Daily</i>
<i>George's Quay (Tara Street Station)</i>	135141	4:05
<i>Aston Quay (O'Connell Bridge &amp; Temple Bar)</i>	325	4:11
<i>Wellington Quay (Temple Bar &amp; Dublin Castle)</i>	312	4:28
<i>Merchant's Quay (Christ Church Cathedral)</i>	1444	4:30
<i>Usher's Quay (The Liberties &amp; Guinness Storehouse)</i>	10997	4:31
<i>Heuston Station (Phoenix Park &amp; Dublin Zoo)</i>	4425	4:32
<i>Arran Quay (Smithfield &amp; Jameson Distillery)</i>	7453	4:33

<i>Location</i>	<i>Stop/Zone</i>	<i>Daily</i>
<i>Ormond Quay Upper (Capel Street)</i>	1479	4:35
<i>Eden Quay (O'Connell Street &amp; Marlborough Street)</i>	297	4:40
<i>Custom House Quay (Gardiner Street &amp; Connolly Station)</i>	135271	4:43
<i>Dublin Airport (Terminal 2)21</i>	21	4:49
<i>Dublin Airport (Terminal 1)1</i>	1	4:56

Fonte: Adaptada de DUBLIN, s. d.

Obviamente que, nesse exemplo, você poderá simplesmente procurar o horário, sem necessidade de recursos computacionais. Mas as coisas nem sempre são simples.

Veja a Figura a seguir, que apresenta a timetable do 782, com paradas e horários divididos por dias da semana. Embora exija um pouco de esforço, ainda é possível localizar os horários desejados sem grandes dificuldades. Mas, e se precisássemos realizar essa busca utilizando o computador? Ou então, imagine que você, Analista de Sistemas do time, recebeu do Setor de Planejamento a missão de ajudar o time *front-end* a realizar a mudança desses horários no website da Empresa. A tarefa certamente não seria trivial, concorda? Como as expressões regulares poderiam ser úteis nesse caso?

**782**

# Dublin Airport to Dublin City Centre & Heuston Station

D	D	D	D	D	SSu	M-F	D	D												
0405	0505	0535	0605	0635	0705	0705	0735	0735	0805	0805	0835	0835	0905	0905	0935	0935	1005	1005	1035	1105
0411	0511	0541	0611	0641	0710	0711	0740	0741	0810	0811	0840	0840	0910	0910	0940	0940	1010	1010	1040	1110
0428	0528	0602	0632	0702	0735	0739	0805	0814	0835	0849	0905	0913	0933	0940	1005	1006	1035	1035	1105	1135
0430	0530	0605	0635	0705	0738	0742	0808	0817	0838	0852	0908	0916	0936	0943	1008	1009	1038	1038	1108	1138
0431	0531	0607	0637	0707	0740	0744	0810	0819	0840	0858	0910	0922	0940	0949	1010	1015	1040	1044	1110	1140
0432	0532	0608	0638	0708	0741	0745	0811	0820	0841	0859	0911	0923	0941	0950	1011	1016	1041	1045	1111	1141
0433	0533	0609	0639	0709	0742	0746	0812	0821	0842	0900	0912	0924	0942	0951	1012	1017	1042	1046	1112	1142
0435	0535	0611	0641	0711	0744	0748	0814	0823	0844	0902	0914	0926	0944	0953	1014	1021	1044	1048	1114	1144
0440	0540	0616	0646	0716	0749	0753	0819	0828	0849	0911	0919	0935	0949	1002	1019	1030	1049	1057	1119	1149
0443	0543	0619	0649	0719	0751	0756	0821	0833	0851	0916	0921	0939	0951	1005	1021	1038	1051	1059	1121	1151
0449	0549	0624	0654	0724	0755	0803	0825	0843	0855	0926	0925	0946	0955	1011	1025	1037	1055	1103	1125	1155
0456	0556	0631	0701	0731	0801	0811	0831	0851	0901	0936	0931	0956	1001	1021	1031	1046	1101	1111	1131	1201

**Figura 1 – Busca de texto utilizando Expressões Regulares**

Fonte: Reprodução

**#ParaTodosVerem:** A imagem apresentada é um recorte de *Our Dublin Airport stops*. Disponível em <https://www.dublinexpress.ie/>. Com o fundo azul-escuro e letras em branco, o texto 782 está em destaque com um azul mais claro e texto branco. O título principal "Dublin Airport to Dublin City Centre & Heuston Station" está em branco. Abaixo existe uma tabela com os dias da semana nas colunas e as paradas do ônibus 782 no eixo x. A tabela tem as dimensões na cor azul e texto branco e o conteúdo com linhas azul-claro e escuro e texto branco. Fim da descrição.

Inicialmente, é preciso esclarecer que não há necessidade de conhecer Programação para aplicar uma REGEX. Basta que a ferramenta com a qual estamos trabalhando permita sua utilização. Com alusão ao cenário do viajante, basta procurar os embarques realizados entre 4h30 e 4h35 e podemos ajudá-lo usando o termo '^0[0-9]3[0-9]!'.

Quanto à expressão apresentada, até o final da disciplina você terá aprendido o significado dela. Por enquanto, observe, na Figura 2, que apenas os horários definidos foram destacados e isso pode ser bastante útil em seu trabalho como analista.

A screenshot of the Visual Studio Code interface. The title bar says "timetable.txt - PIT em ADS I - Visual Studio Code". The left sidebar shows a search bar with the pattern "^0[0-9][0-9]" and a count of "5 results in 1 file · Open in editor". The main editor area displays a list of numbers from 1 to 13, each followed by a series of digits. Several lines are highlighted in yellow, corresponding to the search results. The status bar at the bottom right shows "Ln 15, Col 1 · Spaces: 4 · UTF-8 · LF · Main Text".

**Figura 2 – Busca de texto utilizando Expressões Regulares**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa uma busca de texto utilizando Expressões Regulares. A Figura tem a barra de títulos e a barra de menus na cor cinza. O painel principal tem cor cinza-escuro e texto branco. O texto apresenta uma série de horários entre 04h05 e 09h36. Em destaque, estão os horários entre 4h30 e 4h35. No rodapé, com fundo cinza, existem diversos comandos e uma caixa de busca. Nela, a expressão regular `^0[0-9][0-9]` aparece com diferentes cores. Por fim, a barra de *status* apresenta 5 *matches*, informando que 5 ocorrências foram encontradas. Fim da descrição.

A composição de caracteres e símbolos forma uma sequência, que é interpretada como uma regra e, caso o texto apresente tal equivalência, podemos dizer que deu "*match*". Quem define isso é o compilador/interpretador, quando verificar a combinação entre o conjunto de caracteres e o padrão *pattern* procurado. Note que esse padrão procurado pode variar entre datas, horários, endereços de *e-mail*, números de documentos como CPF e RG, declaração de funções E dados entre `<tags></tags>`, etc.

## Site

Para realizar os testes com as expressões regulares, você poderá utilizar

**diferentes ferramentas. Nesta Disciplina, recomendamos o uso de três ferramentas, pela facilidade de instalação, utilização e disponibilidade de licença gratuita. Elas, no entanto, não são exclusivas. Editores de textos como o MS Word, o Libreoffice Writer e o Google Docs (on-line) têm o recurso. Preste atenção apenas às peculiaridades de cada uma delas.**

#### **Notepad++**

Clique no botão para conferir o conteúdo indicado.

**ACESSE**

#### **Sublime Text**

Clique no botão para conferir o conteúdo indicado.

**ACESSE**

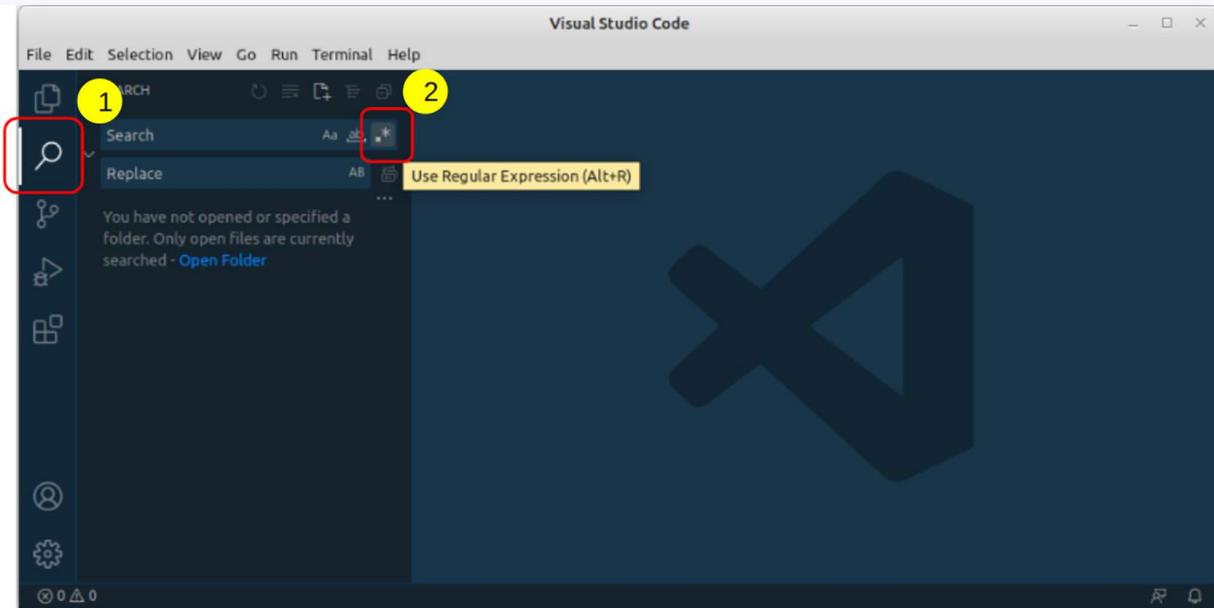
#### **Visual Studio Code (VSCode)**

Clique no botão para conferir o conteúdo indicado.

**ACESSE**

Os exemplos demonstrados nesta Disciplina foram realizados a partir do VSCode. A ferramenta tem o campo de pesquisa utilizando REGEX de forma nativa, mas é preciso habilitá-la. Existem duas formas de realizar buscas desse modo: o primeiro é clicando no painel lateral esquerdo Search e, em seguida, *Use Regular Expression*.

A Figura a seguir representa o processo:



**Figura 3 – Interface do VSCode com o painel search e REGEX ativos**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa o Painel do *Microsoft Visual Studio Code* (VSCode). Ela está dividida entre a barra de títulos com o nome do aplicativo e a barra de menus, ambas na cor cinza e texto na cor preta. A seguir, existem três painéis com o fundo azul-escuro divididos na vertical. O primeiro, localizado na lateral esquerda, conta com uma série de ferramentas. Em destaque, a opção de busca. Na sequência, o painel central de busca apresenta as opções de buscar e substituir. O destaque nesse painel está na opção de habilitar as expressões regulares. O painel principal, mais a direita, está vazio, demonstrando que não há nenhum documento aberto. Fim da descrição.

Esse mesmo recurso pode ser acessado por meio do menu *Edit/Find in files*.

Ainda no menu *Edit*, a segunda forma de criar seus padrões deve ser realizada com um documento aberto. Você poderá navegar até o menu *Edit/Find* ou simplesmente pressionar *CTRL + F*.

**Figura 4 – Busca no VSCode com o REGEX habilitado**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa um documento HTML aberto no *Microsoft Visual Studio Code (VSCode)*. A imagem tem fundo azul-escuro e letras coloridas. Nela, existe um destaque para a utilização das expressões Regulares, habilitada por meio do Alt + R, disponível na caixa de procura. Fim da descrição.

Voltando ao exemplo do telefone, quando trabalhamos com sistemas de informática, é bastante comum que os números apareçam no formato sequencial, sem qualquer espaço ou símbolo. Nesse caso, poderíamos buscá-lo com a REGEX [0-9]. Ela corresponde a qualquer dígito encontrado no intervalo de 0 a 9. Os colchetes não são correspondidos literalmente porque são tratados especialmente como metacaracteres. Um metacaractere tem um significado especial em expressões regulares e é reservado.

## Metacaracteres

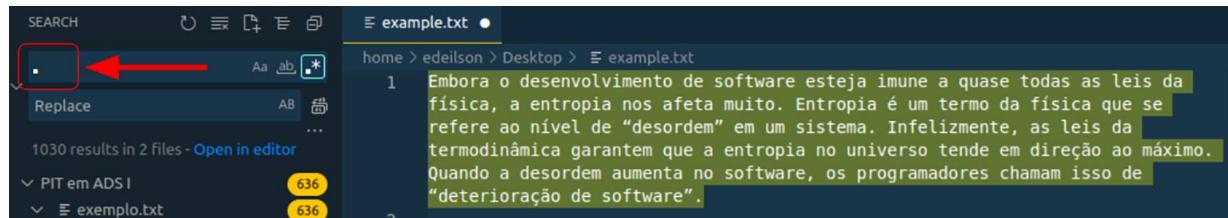
Formada pelo agrupamento entre simbolos chamados de metacaracteres e caracteres literais, uma expressão pode ser considerada uma regra pois, diante das condições estabelecidas, retorna somente os resultados positivos. Os metacaracteres têm funções especiais e não podem ser usados

como um caractere literal. Caso esse seja o objetivo, o metacaractere deve ser precedido por uma barra invertida (\).

Também chamamos esse recurso de escape. ele retorna o valor literal do elemento à sua direita. Veremos diversos metacaracteres, como '.', '\*', '^', '\$', 'l', 'l' etc. Todos eles têm alguma função especial dentro das expressões regulares e quando não estivermos nos referindo a essas expressões, podemos simplesmente utilizar o escape antes deles. Por exemplo: '\.', '\\*', '\^', '\\$', '\l', '\L', etc.

## Metacaracteres Tipo Representante

Por exemplo, em uma busca utilizando REGEX, o ponto final (.) é um caractere curinga e corresponde a todos os caracteres. Ao proceder utilizando esse caractere de forma exclusiva, obteremos como resultado todos os caracteres da *string*, sendo exibidos um por um como decorrência da busca. Ao adicionarmos uma barra invertida antes desse metacaractere, teremos outro resultado. O *match* é feito apenas quando um ponto final for encontrado na cadeia de caracteres.



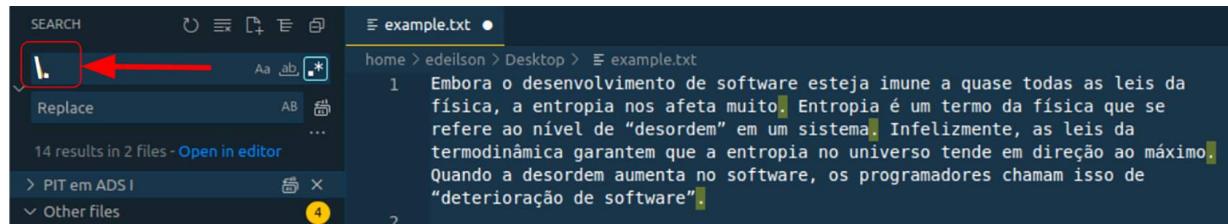
**Figura 5 – Match entre o ponto final e os caracteres de texto**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa um documento de texto aberto no *Microsoft Visual Studio Code (VSCode)*. A imagem tem fundo azul escuro e letras brancas. Nela, existem dois painéis. Ao lado esquerdo, o destaque tem uma seta vermelha apontando para a esquerda, especificamente, para a caixa de busca. No painel central, o texto está todo em destaque e, com fundo na cor verde, significa que houve match entre a REGEX pesquisada e todo o texto. Fim da descrição.

Além do ponto final, que representa todos os caracteres, há opções de representar um ou mais caracteres por meio dos colchetes (ll). De forma específica os colchetes são usados para criar classes

de caracteres, podemos ter uma lista '[..]' e lista negada '[^..]'.



**Figura 6 – Match entre a REGEX '\.' e os caracteres do texto**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa um documento de texto aberto no *Microsoft Visual Studio Code (VSCode)*. A imagem tem fundo azul-escuro e letras brancas. Nela, existem dois painéis. Ao lado esquerdo, o destaque tem uma seta vermelha, apontando para a esquerda, especificamente para a caixa de busca. No painel central, os pontos finais estão em destaque e com o fundo na cor verde, significa que houve *match* entre a REGEX pesquisada e os pontos. Fim da descrição.

Junto com o ponto final, as listas formam uma categoria de representantes. Há, ainda, o grupo de quantificadores, âncoras e alguns outros, mas, por hora, vamos falar um pouco mais sobre as listas.

Considere a Figura 2 apresentada mais cedo, caso fosse necessária uma reorganização dos horários com os seguintes critérios:

- As partidas devem ser realizadas a cada 30 minutos;
- Partidas realizadas entre hora cheia (00) e 29 min. devem mudar para 15 min.;
- Partidas realizadas entre 30 e 59 minutos devem mudar para 45.

Assim, caso a partida seja às **04:22**, o novo horário será **04:15**, em outra situação com partida às **08:55**, o horário deve ser ajustado para **08:45**. Note que o problema apresenta um desafio interessante e,

avaliando o cenário apresentado, minutos preciosos certamente seriam gastos nesta tarefa. Como podemos utilizar a REGEX neste caso?

Inicialmente, adicionamos os ':' (dois pontos) para definir horas e minutos, acreditamos que isso facilite o entendimento. Na sequência, precisaremos buscar minutos e suas variações entre 0 e 2 para o primeiro algarismo, com a expressão '[0-2]'. Note que o '-' (hífen) representando o intervalo entre os valores foi colocado. A ausência desse operador definiria a busca exclusiva dos termos: 0 ou 2.

Um breve exemplo a partir desse segundo argumento seria a busca textual da palavra 'não' escrita sem o acento. A instrução '**nlaão**' traria ambos os resultados, com ou sem acento. Note que uma REGEX pode ser construída com diferentes intervalos numéricos e textuais. Para saber o que cada intervalo contempla é preciso estar atento(a) à Tabela ASCII, pois ela fornece as diretrizes sobre o conjunto de caracteres presentes em uma sequência.

Por fim, precisaremos do complemento que pode ser qualquer dígito entre 0 e 9. Pensando na descrição anterior, teríamos duas possibilidades, '**[0123456789]**' ou '**[0-9]**', sendo o segundo menor, mais intuitivo e mais interessante para essa situação. Assim, temos '**:[0-2][0-9]**'. Essa, no entanto, não é a única forma de realizar uma busca por dígitos decimais. O comando '**\d**' é relativo a qualquer dígito decimal entre 0 e 9. Com isso, poderíamos usar '**:[0-2][\d]**' para obter o mesmo resultado.

A partir do padrão formado, podemos facilmente buscar o resultado inverso, ou seja, partidas com horários entre 30 e 59 min. Uma forma de fazer isso seria modificar a expressão anterior com uma lógica inversa, ou seja, negar o termo anterior pode trazer os valores procurados. A nova expressão ficaria '**:[^0-2][\d]**'.

Time Interval	Count
04:05 - 05:35	120
05:35 - 06:35	120
06:35 - 07:35	120
07:35 - 08:35	120
08:35 - 09:35	120

## Figura 7 – Microsoft Visual Studio

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa um documento de texto aberto no *Microsoft Visual Studio Code (VSCode)*. A imagem tem fundo azul-escuro e letras brancas. Nela existem dois painéis. Ao lado esquerdo, a caixa de busca com a expressão ':[0-2][0-9]'. No painel central, o texto em destaque é, com o fundo na cor verde, apresenta minutos em uma Tabela de horários, que significa que houve *match* entre a REGEX pesquisada e os minutos destacados. Fim da descrição.

---

Como dito, as expressões regulares consideram a Tabela ASCII e, portanto, consideram as diferenças entre localidades. Isso resulta em grandes diferenças realizadas em buscas nos documentos escritos na Língua Portuguesa e Inglesa. O alfabeto norte-americano não tem caracteres acentuados como 'áéóõç', 'áéíóú' ou 'ÁÉÍÓÚ'. Logo, a instrução '[a-z]' não retornaria os caracteres minúsculos.

Para a nossa sorte, alguns artifícios foram criados para auxiliar o trabalho com listas. Sobretudo, quando pensamos na localidade do sistema, as classes denominadas POSIX operam como um padrão para determinar interfaces comuns entre sistemas operacionais. Embora sejam declaradas entre dois pontos e colchetes, é importante lembrar que esses colchetes não se referem às listas.

Vejamos algumas:

- **[:upper:]** - busca letras maiúsculas, equivalente a [A-Z];
- **[:lower:]** - busca letras minúsculas, equivalente a [a-z];
- **[:alpha:]** - busca letras maiúsculas e minúsculas, equivalente a [A-Za-z];
- **[:alnum:]** - busca letras e números, equivalente a [A-Za-z0-9];
- **[:digit:]** - busca letras e números, equivalente a [0-9];
- **[:punct:]** - busca sinais de pontuação, equivalente a [! ? : ... ].

Para contextualizar a utilização de algumas dessas classes, pensemos em uma nuvem de palavras na qual os termos mais comuns são apresentados com maior destaque. A contagem das palavras apresentadas exige que todas as pontuações sejam removidas e com isso a instrução '`[[:punct:]]`' seria o suficiente para localizar todas as ocorrências e então removê-las. Outra forma de atender a solução seria buscar todos os caracteres alfanuméricos, ou seja, letras e números e, então, negar a lista dessa forma '`[^[:alnum:]]`'.

Embora existam outras classes, nesta Disciplina, iremos nos concentrar nas que foram apresentadas até agora. Você perceberá que elas são suficientes para cobrir grande parte das combinações e para iniciar suas REGEX de forma muito ampla. Além disso, esse é um convite para que você acesse os Materiais Complementares e aprofunde o conhecimento no assunto.

## Metacaracteres Tipo Quantificador

Em uma sequência de caracteres, é comum que haja repetições ao longo do documento e quantificar essas ocorrências é fundamental para criarmos REGEX. A categoria de metacaracteres do tipo quantificador permite verificar o número de repetições entre as entidades. Iniciando pelo operador simbolizado pelo ponto de interrogação '?', ele é o chamado opcional e sempre dará *match* se houver ou der ocorrências da entidade anterior. A instrução '`codigo?`' retornaria tanto a palavra 'código' quanto a palavra 'códigos' no plural.

Para buscar as repetições também podemos utilizar o asterisco '\*', que retorna o resultado positivo com zero ou mais ocorrências e o sinal de positivo '+', que dá *match* quando localiza uma ou mais ocorrências. Considerando uma página *web* com código HTML e suas *tags* de abertura e fechamento, a instrução '`</+html`' daria *match* apenas com a *tag* de fechamento. Isso, porque exige que um ou mais atributos '/' sejam contemplados, enquanto '`</*html`', em que não existe a ocorrência da entidade, retorna positivo para ambas as *tags*.

Para concluir os metacaracteres quantificadores, temos as chaves '{}'. Com elas, podemos especificar exatamente quantas repetições do elemento anterior estamos procurando. É possível buscar o número exato de ocorrências utilizando '{valor}', ou intervalos com '{min,max}' e '{min,}'. Faça o teste: qual seria o *match* da REGEX '`\d{4}`' para o número de telefone: (99) 99999-9999?

## Metacaracteres Tipo Âncora

Os metacaracteres do tipo âncora marcam uma posição específica na linha. As âncoras comumente usadas são o acento circunflexo '^' e o cifrão '\$'. Eles correspondem, respectivamente, ao início e ao final de uma linha.

Alguns dos exemplos são:

- **^[AO]** - localizar linhas iniciadas pelos artigos definidos;
- **[0-9]\$** - localiza linhas dígitos ao final da linha e não da palavra;
- **.;|\$** - localiza linhas que encerram com um ponto final ou ponto e vírgula.

Observe que os metacaracteres '^' e '\$' operam nas linhas. Caso seja necessário buscar termos específicos, precisaremos utilizar a borda '\b'. Com ela, poderemos buscar palavras específicas por seu prefixo ou sufixo.

Vejamos:

**Quadro 1**

Expressão	Match
dia	dia, diafragma, melodia, radial, bom-dia!
\b dia	dia, diafragma, bom-dia!
dia\b	dia, melodia, bom-dia!

Expressão	Match
\bdia\b	dia, bom-dia!

Cabe lembrar, ainda, que o acento circunflexo também pode ser utilizado dentro das listas. Nesse caso, ele tem a característica de negação, ou seja, enquanto a expressão '^[o-g]' buscaria linhas iniciadas em números, '^[^o-g]' faria exatamente o contrário.

## Outros Metacaracteres

Quanto trabalhamos com listas, sabemos que a expressão '^[AO]' representa uma busca que dará *match* com algum dos artigos, A ou O, no início da linha. Essa comparação também poderá ser realizada entre listas com o operador barra vertical '|'. Dessa forma, poderíamos buscar os artigos em seu formato literal com a instrução 'A|O' ou, então, buscar termos entre diferentes listas. Considere a Tabela de Horários apresentada anteriormente. Caso fosse necessário buscar partidas às 7h30 ou às 9h30, a expressão '**\d{2}|\d{2}:30**' resolveria o problema.

Esse último exemplo dá dimensão do poder dos metacaracteres utilizados em conjunto. Pode-se trabalhar, ainda, em grupos de caracteres, utilizando os parênteses de abertura e fechamento '()' . Esses blocos ajudam a buscar termos específicos como, por exemplo: '**a(b|c)d**', que pode dar *match* com 'abd' ou 'acd', enquanto '**ab|cd**' retornará 'ab' ou 'cd'. Para ficar mais claro, pense na instrução '**(super l hiper) mercado**'. Quais seriam os resultados?

Além de agrupar um conjunto de caracteres, os parênteses também podem ser úteis para realizarmos substituições. Os grupos são naturalmente identificados em ordem numérica e, por isso, na sequência '**([0-9]{2})([0-9]{2})**' seria identificada como \$1\$2. Considerando a *timetable* do 782, podemos facilmente adicionar os dois pontos substituindo a expressão por '**\$1:\$2**'.

	04:05	05:05	05:35	06:05	06:35	07:05	07:05	07:35	07:35
1	04:11	05:11	05:41	06:11	06:41	07:10	07:11	07:40	07:41
2	04:28	05:28	06:02	06:32	07:02	07:35	07:39	08:05	08:14
3	04:30	05:30	06:05	06:35	07:05	07:38	07:42	08:08	08:17
4	04:31	05:31	06:07	06:37	07:07	07:40	07:44	08:10	08:19
5	04:32	05:32	06:08	06:38	07:08	07:41	07:45	08:11	08:20
6	04:33	05:33	06:09	06:39	07:09	07:42	07:46	08:12	08:21
7	04:35	05:35	06:11	06:41	07:11	07:44	07:48	08:14	08:23
8	04:40	05:40	06:16	06:46	07:16	07:49	07:53	08:19	08:28
9	04:43	05:43	06:19	06:49	07:19	07:51	07:56	08:21	08:33
10	04:49	05:49	06:24	06:54	07:24	07:55	08:03	08:25	08:43
11	04:56	05:56	06:31	07:01	07:31	08:01	08:11	08:31	08:51
12									

## Figura 8 – Tabela de horários após receber a adição de dois pontos

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa um documento de texto aberto no Microsoft Visual Studio Code (VSCode). A imagem tem fundo azul escuro e letras brancas. Nela, existem dois painéis. Ao lado esquerdo, a caixa de busca com a expressão '([0-9]{2})([0-9]{2})' e a substituição por '\$1:\$2'. No painel central, o texto apresenta uma tabela de horários. Fim da descrição.

---

Ao longo desta Disciplina, vimos o símbolo '\d', uma barra invertida acompanhada de uma letra que representa um valor numérico. Com ele, podemos localizar valores entre 0 e 9 dentro de expressões aumentando a abrangência desse recurso. Ele faz um conjunto de metacaracteres do tipo barra-letra e aumenta substancialmente o poder das expressões regulares.

Cada letra representa um comando diferente. Por exemplo, se for necessária uma busca por caracteres numéricos, já sabemos que o '\d', que também pode ser representado como lista [0-9], é uma opção. E se precisássemos rastrear os valores não numéricos, é simples, utilizamos o '\D', ou seja, existe diferença entre letras maiúsculas e minúsculas. Experimente buscar os espaços em branco com '\s' e preenchidos com o '\S', ou, então, '\w' que corresponde a qualquer caractere alfanumérico e '\W' que corresponde a qualquer caractere não alfanumérico.

Isso não significa que todas as letras do alfabeto tenham sido utilizadas. É preciso ter em mente que cada Linguagem de Programação, sua interface de desenvolvimento (IDE) ou aplicação que permite a utilização de REGEX tem sua própria notação. Além disso, podemos fazer equivalências entre os

metacaracteres barra-letra e as classes POSIX. A principal vantagem de utilizar o barra-letras é que eles podem ser usados fora das listas.

**Tabela 2 – Equivalência entre metacaracteres e classes**

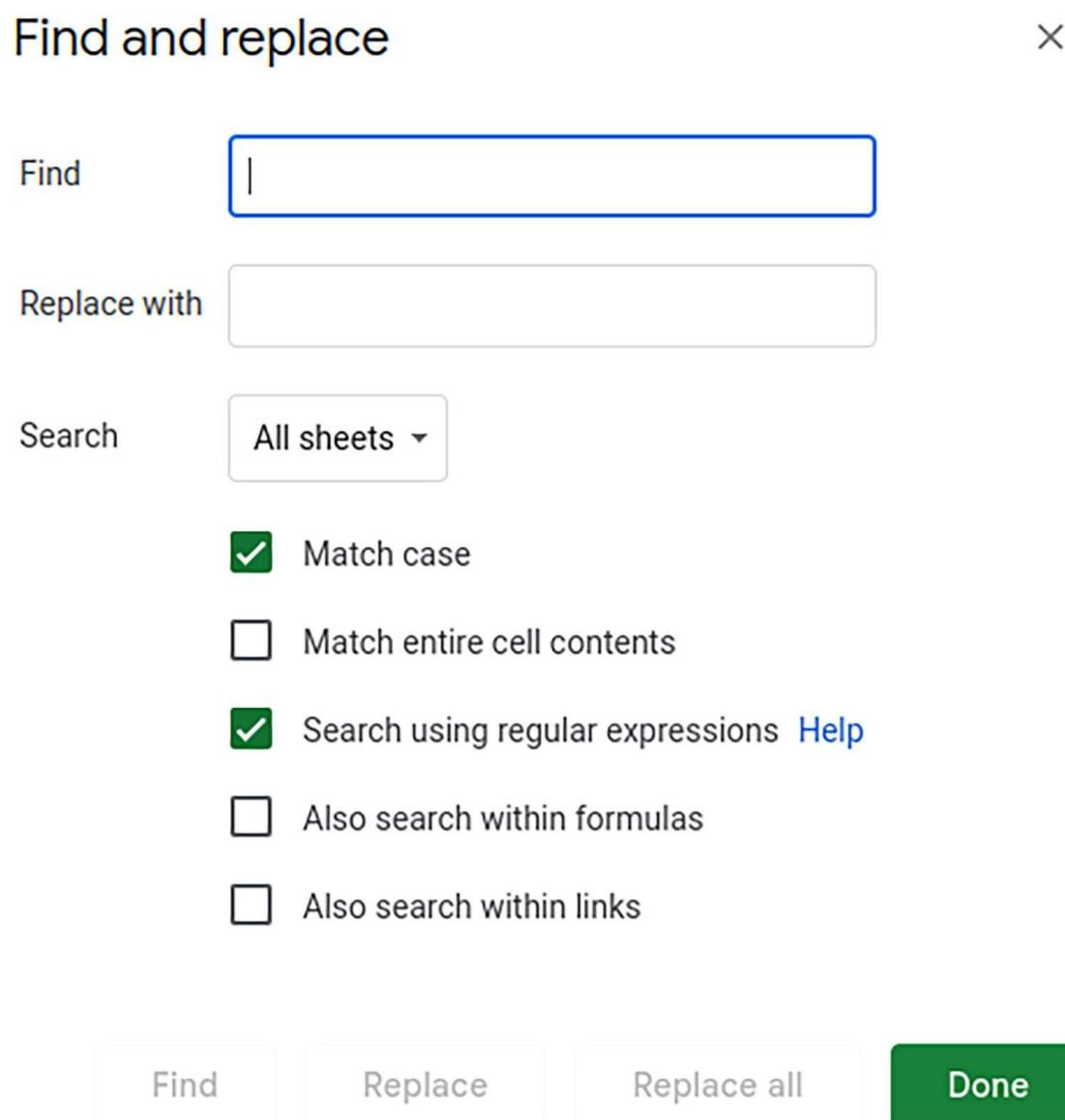
Barra-letra	Equivalente POSIX	Significa
\d	[:digit:]	Dígito
\D	[^[:digit:]]	Não-dígito
\w	[:alnum: ]_	Palavra
\W	[^[:alnum: ]_]	Não-palavra
\s	[:space:]	Branco
\S	[^[:space:]]	Não-branco

Fonte: Adaptada de JARGAS, 2012, p. 68

## Expressões Regulares com Google Planilhas

Muitas linguagens, ferramentas e plataformas utilizam expressões regulares e sua utilização é importante para todas elas. Conhecer REGEX certamente ajudará os profissionais em Análise e Desenvolvimento de Sistemas a resolver muitos problemas em qualquer área de atuação. Nesta Disciplina, considerando as diferentes experiências e habilidades dos estudantes, optamos por avançar com a utilização da Google Planilhas. Além de não ser necessária a instalação e a configuração, a ferramenta está disponível *on-line*. Dessa forma, todos poderão acessá-la sem grandes esforços e utilizar sua lógica em qualquer linguagem de programação.

Inicialmente, podemos utilizar a função *Find and Replace*, disponível em *Google Docs*, *Google Slides* e *Google Sheets*. Com um documento aberto, navegue até o menu Editar e clique em Localizar e Substituir. Ao lado de "Localizar", digitamos a expressão procurada e caso haja a necessidade de substituição será preciso preencher o novo termo no campo "Substituir". Para ativar a REGEX, clicamos em "Pesquisar usando expressões regulares".



**Figura 9 – Localizar e Substituir**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura representa o comando Localizar e Substituir e pode ser encontrada no *Google Docs*, no *Sheet* e no *Slides*. A Figura tem fundo branco e texto da cor preta. O título "Find" and "Replace" é acompanhado por campos de texto em

branco. O campo "Find" está em destaque e aguardando para que um texto seja digitado. A opção Buscar tem um menu dropdown, com o termo "Todas as planilhas" selecionado. Em uma lista de opções, os campos "Correspondência exata" e "Busca utilizando expressões regulares" estão marcados em verde e sinalizados para a busca. No final da imagem, existem quatro botões: localizar, Substituir e Substituir, tudo esmaecidos. O botão "Feito" está em destaque e com o fundo verde. Fim da descrição.

---

Na ferramenta de planilha, as expressões regulares são úteis para resolver problemas com *strings* de texto por meio da combinação de padrões e, por meio dela, podemos encontrar nomes ou números de telefone entre uma grande quantidade de dados, validar endereços de *e-mail*, extrair URLs, renomear nomes de arquivos etc.

Para trabalhar com REGEX no âmbito das planilhas, a Google disponibiliza três funções: REGEXMATCH, REGEXEXTRACT e REGEXREPLACE. Com ela, podemos localizar e substituir *strings* de texto, incluindo caracteres, números e palavras ou padrões por expressões regulares documentos, planilhas e *slides*.

- REGEXMATCH confirmará se encontrou o padrão no texto;
- REGEXEXTRACT extrairá o texto que corresponde ao padrão;
- REGEXREPLACE substituirá o texto que corresponde ao padrão.

A função REGEXMATCH retorna TRUE se corresponder ao padrão fornecido for localizado em qualquer lugar do texto e FALSE caso não haja correspondências no texto. Vejamos uma breve lista com livros da série Vaga-Lume, uma coleção que fez a cabeça de muitos jovens e adolescentes entre os anos 1970 e 2000. No primeiro exemplo, a expressão '=REGEXMATCH(A2, "[0-9]")' retornou TRUE apenas para as células cujo texto continha algum valor numérico:

	A	B
1	Título do livro	REGEXMATCH
2	1973 – Éramos seis (Maria José Dupré)	? =REGEXMATCH(A2, "[0-9]")
3	O Mistério do 5 Estrelas (Marcos Rey)	TRUE
4	A Turma da Rua Quinze (Marçal Aquino).	FALSE
5	Aventura no Império do Sol (Silvia Cintra Franco, 1989)	TRUE
6	1992 – Um rosto no computador (Marcos Rey)	TRUE

**Figura 10 – Lista com livros da série Vaga-Lume**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura apresenta uma Tabela de duas colunas e sete linhas. A primeira linha conta com os cabeçalhos da coluna tem fundo preto e cor branca. As colunas A e B têm os respectivos textos: Título do livro e REGEMAX. O restante das linhas contém os títulos dos livros na coluna A e o termo TRUE ou FALSE. Eles referem-se ao resultado da expressão REGEXMATCH apresentada em destaque na célula B2. A linha 4 é a única em destaque com o fundo amarelo e o resultado FALSO. Fim da descrição.

---

Considerando esse mesmo exemplo, você poderá extrair o ano de publicação dos livros. Para isso, foi criada uma coluna chamada ano e utilizada a função REGEXEXTRACT. Observe que o metacaractere dígito seguido do sinal de positivo retornou valores individuais para as células que continham valor numérico e o erro #N/A para a única que não continha. Embora não seja o foco, saiba que a *Google Sheet* fornece uma função para tratar esse tipo de erro ISNA() que, quando combinada a outras funções, poderia dar um tratamento mais adequado ao problema. Experimente usar a expressão '=IF(ISNA(REGEXEXTRACT(A4,"\\d+")),FALSE)'.

	A	B
1	Título do livro	Ano
2	1973 – Éramos seis (Maria José Dupré)	? =REGEXEXTRACT(A2, "\d+")
3	O Mistério do 5 Estrelas (Marcos Rey)	5
4	A Turma da Rua Quinze (Marçal Aquino).	#N/A
5	Aventura no Império do Sol (Silvia Cintra Franco, 1989)	1989
6	1992 – Um rosto no computador (Marcos Rey)	1992

**Figura 11 – Lista com livros da série Vaga-Lume**

Fonte: Reprodução

**#ParaTodosVerem:** A Figura apresenta uma Tabela de duas colunas e sete linhas. A primeira linha conta com os cabeçalhos da coluna tem fundo preto e cor branca, as colunas A e B têm os respectivos textos: Título do livro e Ano. O restante das linhas contém títulos de livros na coluna A e seu ano de publicação na coluna B. Eles se referem ao resultado da expressão REGEXEXTRACT apresentada em destaque na célula B2. A linha 4 é a única em destaque, com o erro #N/A e fundo amarelo. Fim da descrição.

Já mencionamos, mas vale relembrar que existem muitas peculiaridades quando falamos de expressões regulares. Na Google, a fórmula REGEXEXTRACT retorna um atributo de texto, e caso aparecesse alguma continuidade envolvendo o valor numérico, algum tratamento, como no exemplo do erro apresentado, seria necessário.

Além disso, se o texto apresentasse equivalência de dígitos numéricos em diferentes partes do texto, somente a primeira ocorrência retornaria. Nesse caso, o título "O Mistério do 5 Estrelas (Marcos Rey, 1981)" retornaria o valor 5, o que certamente não satisfaz a nossa condição e, por isso, precisa tratamento. Essa será uma ótima forma de praticar o que você aprendeu na Disciplina e, portanto, deixaremos a resposta como uma das atividades propostas.

Ainda nesse cenário, algo bastante comum é a necessidade de substituição de caracteres. Por exemplo, o número 5 de "O Mistério do 5 Estrelas (Marcos Rey, 1981)" pode ser substituído por 'cinco' utilizando: '=REGEXREPLACE(A3,"\\d\\s", "cinco ")'.

Nesse caso, a própria REGEX foi suficiente para localizar e substituir o caractere em questão, mas caso não seja possível fazê-lo em uma única instrução, lembre-se de que a combinação entre funções e expressões regulares será sempre uma opção.

Nesta Disciplina, nem de longe tivemos a intenção de exaurir o assunto. Pelo contrário, esperamos que você possa ter aprendido um pouco mais sobre as expressões regulares, entendida a importância para resolver problemas computacionais e fomentando seu interesse sobre o tema.

Agora é com você: aproveite para acessar os Materiais Complementares e aprofundar o conhecimento no assunto.

# Material Complementar

---

Indicações para saber mais sobre os assuntos abordados nesta disciplina:

---

## LEITURAS

### Compilando Expressões Regulares

Tutorial introdutório sobre expressões regulares em *Python*.

<https://bit.ly/3KZZhNl>

### Expressões Regulares

Uso de expressões regulares em *JavaScript*.

<https://mzl.la/3SSSRsj>

### The POSIX Family of Standards

O Artigo apresenta características da *Portable Operating System Interface (POSIX)* e sua importância para o desenvolvimento de sistemas.

<https://bit.ly/3ZP1SoG>

### Regular Expressions Cheat Sheet

Um guia de referência rápida para expressões regulares (REGEX), incluindo símbolos, intervalos,

agrupamento, asserções e alguns padrões de amostra para você começar.

<https://bitly/3Zl9HGd>

# Situação-Problema 1

---

Caro(a), estudante.

Agora, vamos compreender o cenário que será abordado na primeira situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

## Resolvendo Problemas com Expressões Regulares

Projetos de *software* frequentemente se deparam com situações em que desenvolvedores e analistas se reúnem para buscar soluções simples, mas que resolvem um grande problema. Um exemplo bastante comum é a validação de dados: embora seja uma ação direta do desenvolvedor, a tarefa pode ser facilitada quando outros profissionais colaboram.

Nessa primeira situação, a equipe se deparou com problemas na validação do *e-mail*, sem considerar uma linguagem específica.

Como poderiam realizar essa validação?

## Situação-Problema 2

---

Vamos compreender o cenário que será abordado na segunda situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

Quando o produto de *software* é disponibilizado para diferentes países, é comum que determinados padrões sofram alterações, caso da data no formato norte-americano e brasileiro, por exemplo. Nos Estados Unidos, a data é normalmente precedida pelo ano e tem como sequência o mês e o dia.

Nesse cenário, como as expressões regulares podem ajudar você a transformar as datas no formato 2022-11-30 para o padrão brasileiro?

## Situação-Problema 3

---

---

Por fim, vamos compreender o último cenário, abordado na terceira situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

Sabendo do seu talento como Analista de Sistemas, o Gestor de Projetos solicitou seu auxílio para uma tarefa na *Google Sheet*. A exportação de dados trouxe alguns caracteres problemáticos e por isso a validação da lista com aproximadamente 2000 endereços será necessária.

Agora que conhece as expressões regulares, como você poderia auxiliar o gestor utilizando a ferramenta?

## Problema em Foco

---

Validar o *e-mail* do usuário é algo necessário para toda aplicação que requer um cadastro e em qualquer linguagem. Nesse sentido, as expressões regulares poderão ser bastante úteis e auxiliar o time de desenvolvimento. A sugestão inicial para a primeira situação é que a validação utilize o "@" como referência e que seja dividida em etapas, a primeira para localizar.

O segundo estudo de caso também é fato recorrente em diversos escritórios ao redor do Brasil. Muitas vezes, um documento traduzido, uma planilha da qual fazemos *download* ou a extração de dados de um sistema traz o padrão norte-americano de data. Isso não é um problema, pois, muitas vezes, o objetivo é padronizar o que será armazenado na base. Entretanto, esse pode ser um problema para o utilizador final.

As expressões regulares são bastante úteis e não se limitam a linguagens de programação. Diversas aplicações permitem sua utilização e facilitam o processo de validação, extração e substituição de dados, caso da *Google Sheet*.

## Atividade de Entrega

---

Muito bem, estudante.

Agora que você já leu todas as situações-problema, você pode fazer o download [deste arquivo](#) para realizar a atividade de entrega.

Caso prefira, o arquivo também se encontra no Ambiente Virtual de Aprendizagem.

## Referências

---

FITZGERALD, M. *Introducing Regular Expressions: Unraveling Regular Expressions, Step-by-Step.* [S.l.]: O'Reilly Media, 2012.

JARGAS, A. M. **Expressões regulares:** uma abordagem divertida Aurélio Marinho Jargas. 4. ed. São Paulo: Novatec, 2012.

---

 Muito bem, estudante! Você concluiu o material de estudos! Agora, volte ao Ambiente Virtual de Aprendizagem para realizar a Atividade.