Expresiones regulares (RegEx)

Departamento de Lenguajes y Sistemas Informáticos

Universidad de Sevilla

Daniel Ayala, Carlos Arévalo, José Calderón, Margarita Cruz, Inma Hernández, David Ruiz

Contenidos

Validación de formularios

Todos los datos que introduce el usuario en un formulario web **deben ser validados**, para evitar que llegue a la base de datos información incorrecta que provoque incoherencias, y también para evitar amenazas que pongan en riesgo la seguridad del sistema.

Validación

- Validaciones más comunes:
  - Valores vacíos
  - Tamaños de cadena
  - Números máximos y mínimos
  - **Formato de cadenas**

Por ejemplo…

¿Qué formato debe tener una cadena que represente…?

- Un código postal
- Un DNI
- Un aula de la ETSII

Por ejemplo…

¿Qué formato debe tener una cadena que represente…?

- Un código postal

```
"Cinco dígitos"
```

- Un DNI

- Un aula de la ETSII

Por ejemplo…

¿Qué formato debe tener una cadena que represente…?

- Un código postal

```
"Cinco dígitos"
```

- Un DNI

```
"Ocho dígitos seguidos de una letra mayúscula"
```

- Un aula de la ETSII

Por ejemplo…

¿Qué formato debe tener una cadena que represente…?

- Un código postal

```
"Cinco dígitos"
```

- Un DNI

```
"Ocho dígitos seguidos de una letra mayúscula"
```

- Un aula de la ETSII

```
"Una letra mayúscula, un número del 1 al 4, un punto,
y dos números"
```

Por ejemplo…

¿Qué formato debe tener una cadena que represente…?

- Un código postal

```
"Cinco dígitos"
"\d{5}"
```

- Un DNI

```
"Ocho dígitos seguidos de una letra mayúscula"
"\d{8}[A-Z]"
```

- Un aula de la ETSII

```
"Una letra mayúscula, un número del 1 al 4, un punto,
y dos números"
"[A-Z][1-4]\.\d{2}"
```

Usos

- Una expresión regular define un **patrón** de cadena que puede **buscarse** o **reemplazarse** dentro de un texto. P.ej. `\d{5}`

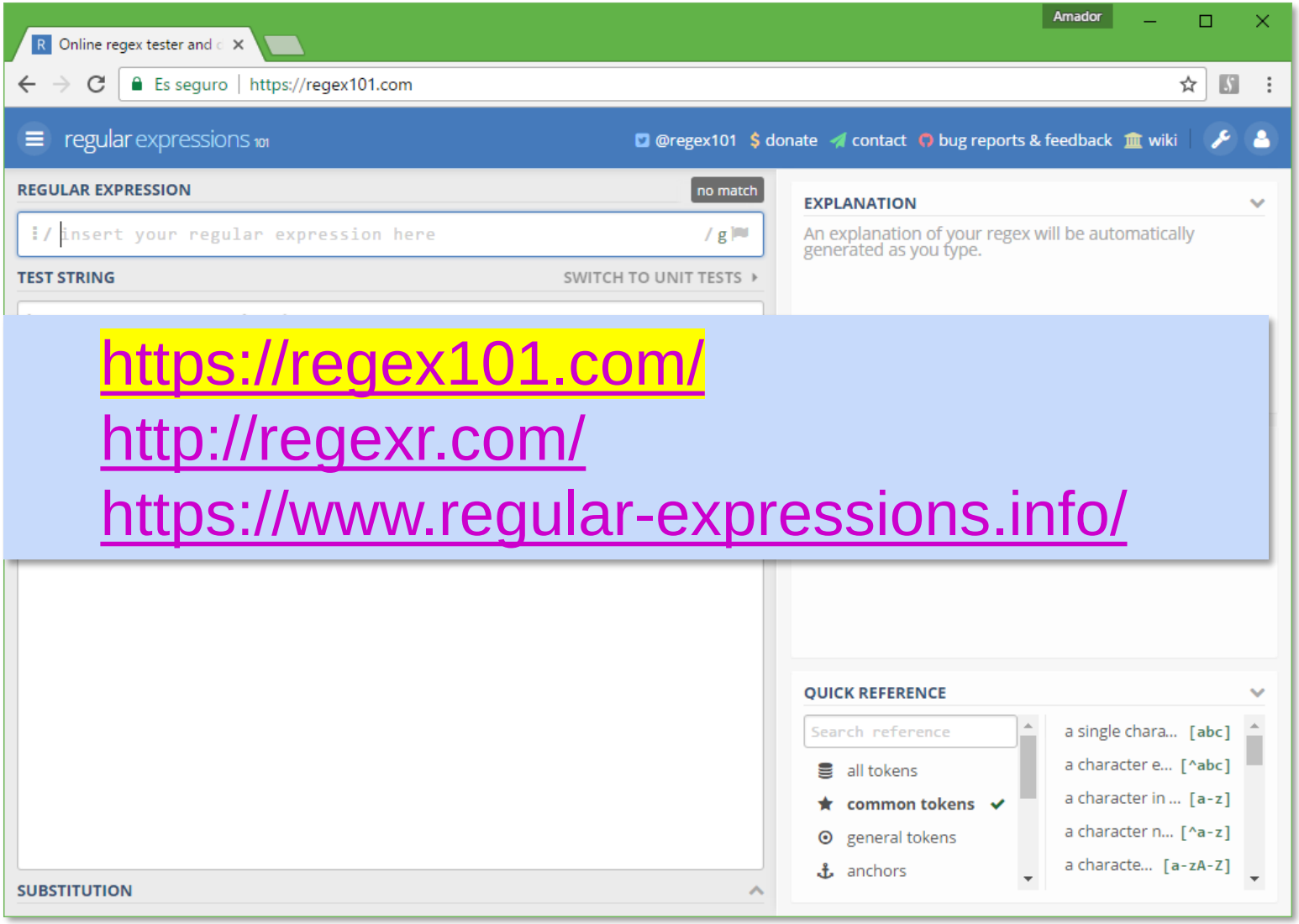
- Comprobar si una cadena **cumple el patrón**
  - `"41012"` ✅
  - `"E.T.S. Informática"` ❌

- Comprobar si una cadena **contiene el patrón** en cualquier posición
  - `"E.T.S. Ingeniería Informática, 41012 Sevilla"` ✅
  - `"Escuela Técnica Superior de Ingeniería Informática"` ❌

Contenidos

1. Introducción
2. **Sintaxis de expresiones regulares**
   - **Patrones (literales, metacaracteres, rangos)**
   - **Cuantificadores**
   - **Grupos**
   - **Modificadores**
3. Ejemplos prácticos

Herramientas online



Nuestro ejemplo: Gazpacho

Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.

https://regex101.com/r/ymBX4q/2

Sintaxis de Expresiones Regulares

```
/patrón/flags
```

Caracteres reservados

- Equivalentes a las palabras reservadas de los lenguajes de programación, tienen significados concretos
- Son los siguientes:
  - `.` representa cualquier carácter
  - `+ * ?` son cuantificadores
  - `^ $` representan principio/final de la cadena
  - `( ) [ ] { }` sirven para definir grupos/rangos/cuantificadores
  - `|` sirve para definir opciones
  - `\` se utiliza para meta-caracteres y escape

Sintaxis del patrón

- El patrón consiste en una combinación de identificadores y cuantificadores
- Tipos de identificadores:
  - Literales
  - Meta-caracteres
  - Rangos

Literales

- Cualquier carácter que no sea un carácter reservado (incluido espacios)
- Los caracteres reservados se pueden convertir a literales escapándolos con `\` (p.ej. `\.` representa el carácter `'.'`)

`/a/`

Carácter “a”

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Literales

- Cualquier carácter que no sea un carácter reservado (incluido espacios)
- Los caracteres reservados se pueden convertir a literales escapándolos con `\` (p.ej. `\.` representa el carácter `'.'`)

/ /

Espacio en blanco

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Literales

- Cualquier carácter que no sea un carácter reservado (incluido espacios)
- Los caracteres reservados se pueden convertir a literales escapándolos con `\` (p.ej. `\.` representa el carácter `'.'`)

```
/oliva/
```

Secuencia de caracteres "oliva"

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Meta-caracteres

- Ciertos tipos de caracteres:
  - `.` Cualquier carácter, excepto fin de línea
  - `\w` Cualquier carácter alfanumérico o guión bajo
  - `\d` Cualquier dígito
  - `\s` Espacios (incluye tabulaciones y saltos de línea)
  - `\n` Fin de línea
  - `\t` Tabulador
  - `\b` Principio/fin de palabras
  - `^` Inicio de línea
  - `$` Fin de línea
- Meta-caracteres negativos:
  - `\W` Cualquier carácter **NO** alfanumérico o guión bajo
  - `\D` Cualquier carácter **NO** dígito
  - `\S` Cualquier carácter **NO** espacio

Meta-caracteres

/\w/

Carácteres alfanuméricos

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Meta-caracteres

/\d/

Números

Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.

Meta-caracteres

`/\s/`

Espacios y saltos de línea

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de
agua de la nevera y 50 ml de vinagre de Jerez, triturando todo
en la batidora de vaso o Thermomix. Una vez triturado, pasamos
el gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Meta-caracteres

/ . /

Cualquier carácter

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Meta-caracteres

/\./

Puntos

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Meta-caracteres

```
/\bto/
```

"to" al principio de una palabra

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Meta-caracteres

/to\b/

"to" al final de una palabra

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de
agua de la nevera y 50 ml de vinagre de Jerez, triturando todo
en la batidora de vaso o Thermomix. Una vez triturado, pasamos
el gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Rangos

- Se utilizan corchetes `[ ]` para indicar rangos de caracteres válidos
  - `[abc]`: Encuentra cualquiera de los caracteres dentro de los corchetes (a, b, c)
  - `[^abc]`: Encuentra cualquier carácter que no esté dentro de los corchetes (cualquiera menos a, b, c)
  - `[0-9]`: Cualquier carácter en el rango entre 0 y 9 (dígitos)
  - `[a-z]`: Cualquier carácter en el rango a-z (minúsculas)
  - `[A-Z]`: Cualquier carácter en el rango A-Z (mayúsculas)
  - `[a-zA-Z]`: Cualquier letra
  - `[^a-zA-Z]`: Cualquier carácter que no sea una letra
  - `[x|y]`: Cualquiera de las alternativas separadas por `|`

Rangos

/[aeiou]/

Vocales

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Rangos

```
/[^aeiou]/
```

Cualquier cosa que no sea una vocal

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Rangos

/[0-3]/

Dígitos entre 0 y 3

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Rangos

```
/[a|0-2]/
```

Carácter 'a' o dígito entre 0 y 2

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

¡Ojo con los rangos!

- Si queremos validar que un número está entre 0 y 120, podemos tener la tentación de usar la expresión `/[0-120]/`
- Lo que ve RegEx:

```
[0-120]
```

  - Cualquier carácter en el intervalo '0'-'1' (códigos ascii 48-49)
  - El carácter '2'
  - El carácter '0'
- RegEx siempre trabaja a nivel de **caracteres**, no de palabras
- En realidad, esta expresión sólo hace "match" a los caracteres individuales `'0','1'o'2'`
  - No siempre es necesario usar RegEx, especialmente para restricciones numéricas

Cuantificadores

- Siendo `p` cualquier patrón
  - `p*` Cualquier número de veces `p` (0..N)
  - `p+` Al menos una vez `p` (1..N)
  - `p?` Cero o una vez `p` (0..1)
  - `p{X}` X veces `p`
  - `p{X,Y}` Entre X e Y veces `p` (ambos inclusive)
  - `p{X,}` Al menos X veces `p`

Cuantificadores

/.*/

Cualquier cadena

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Cuantificadores

/\d+/

Cantidades numéricas

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Cuantificadores

```
/\d{3}/
```

Cantidades numéricas de 3 dígitos

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua
de la nevera y 50 ml de vinagre de Jerez, triturando todo en la
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
```

Cuantificadores

```
/\d+ (kg|g|ml)/
/\d+ (k?g|ml)/
```

Cantidades con kg, g o ml (ambas son equivalentes)

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Cuantificadores

```
/[A-Z]\w*/
```

Palabras que empiecen por mayúscula

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

Grupos de captura

- Se pueden utilizar paréntesis `( )` para definir **grupos de captura**, regiones de interés dentro de una expresión regular de las que queremos guardar su contenido
  - Los paréntesis usados para definir los grupos no se buscan literalmente en el patrón, a no ser que se escapen con `\`
  - Ejemplo: capturar pares "clave: valor" usando un grupo para cada elemento: `/(.*): (.*)/`

```
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m

Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
```

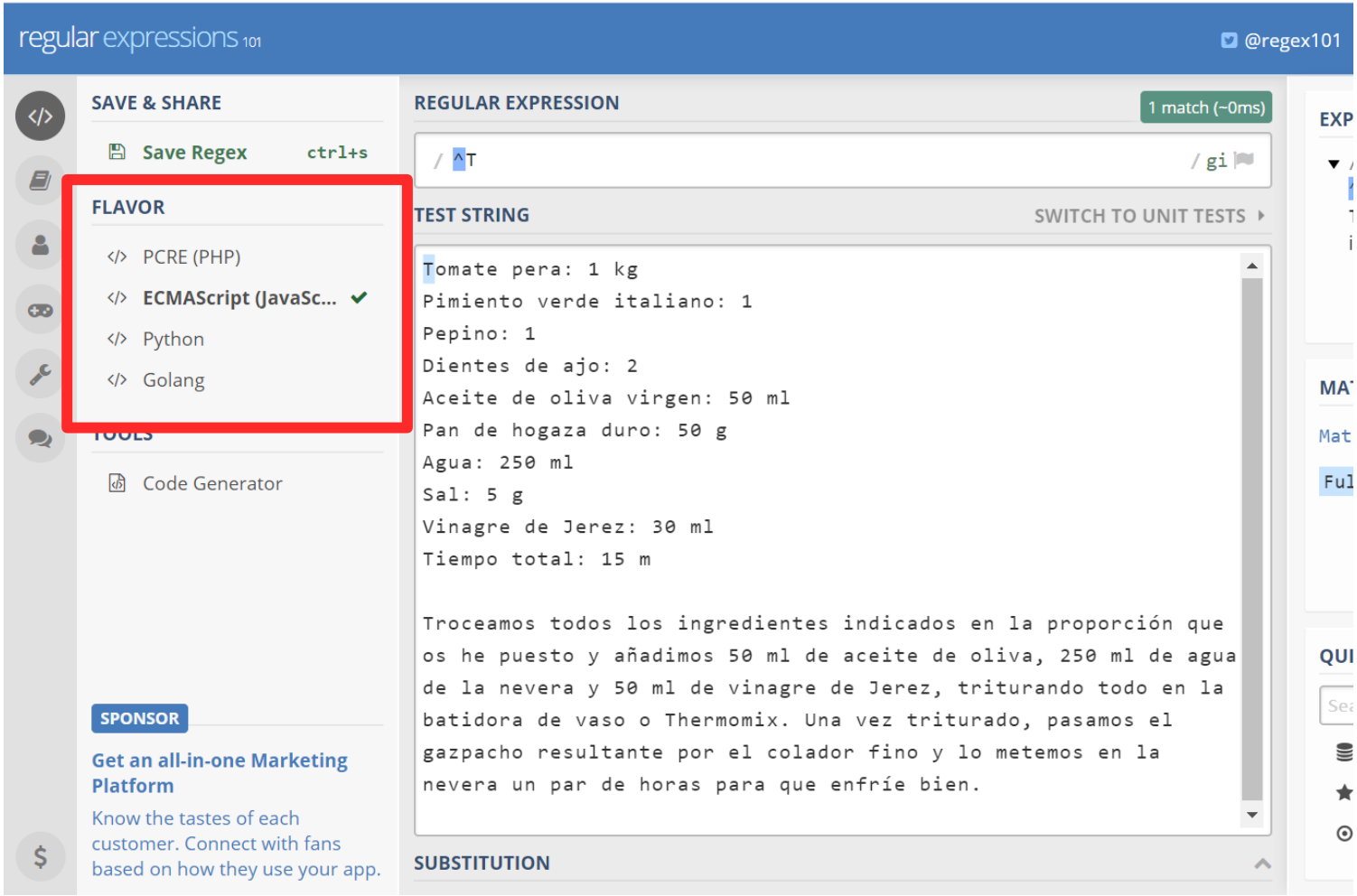
Modificadores

- Los modificadores (*flags*) permiten variar la forma en que se busca el patrón. Ejemplos:
  - **/i (case-insensitive)**: No distingue entre mayúsculas/minúsculas
  - **/g (global)**: Busca todas las apariciones del patrón (si no se indica, se para cuando encuentra el primero)
  - **/m (multiline)**: Afecta al comportamiento de los metacaracteres **$** y **^**. Cuando está activo, se encuentra el patrón no solo al principio y final del texto completo, si no, al principio/final de línea.
  - **\s (single-line)**: Afecta al comportamiento del metacarácter **.** , cuando está activo, hace que también incluya saltos de línea

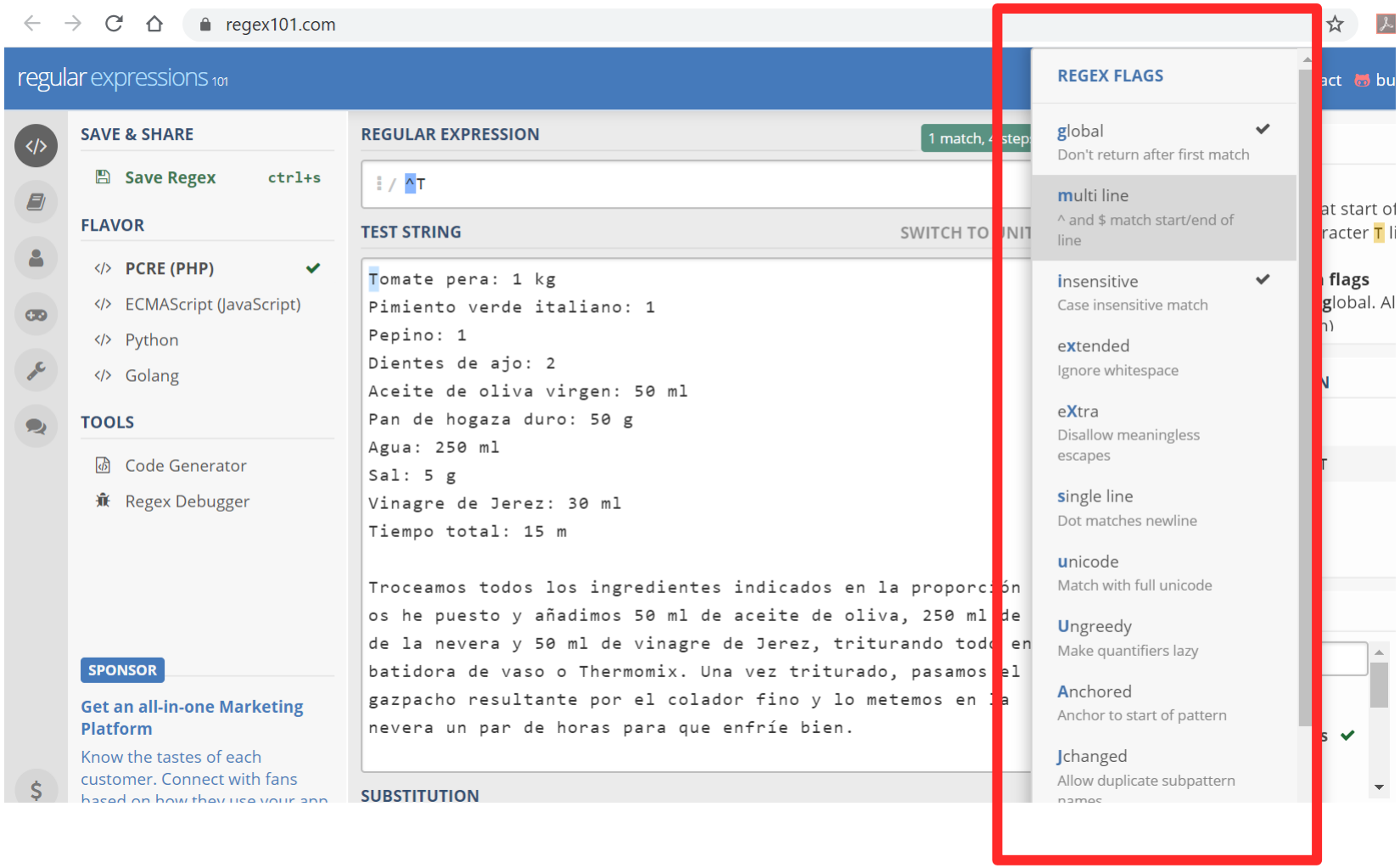
RegExp flavors

- Todos ellos hacen básicamente lo mismo
  - Intentan encontrar la cadena más larga que concuerde con cada expresión regular, empezando por la izquierda
- Cada uno de ellos se basa en un algoritmo determinado y suele estar vinculado a un lenguaje, y de uno a otro pueden variar:
  - Sintaxis
  - Modificadores aceptados
  - Qué se entiende por “cadena más larga”
  - La posibilidad de look-around
  - Optimización

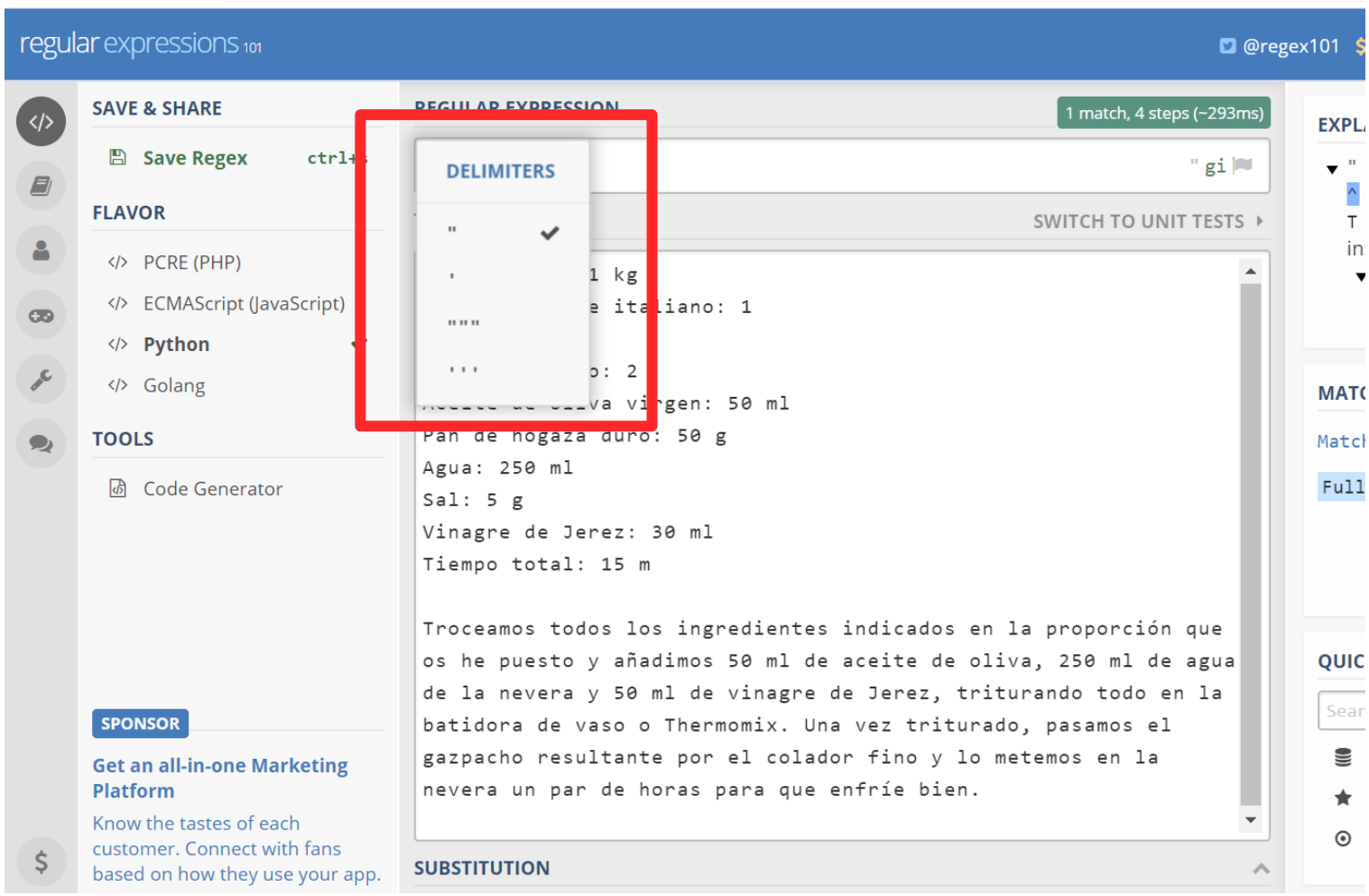
En regex101


regular expressions 101
@regex101
SAVE & SHARE
Save Regex ctrl+s
FLAVOR
PCRE (PHP)
ECMAScript (JavaSc...
Python
Golang
TOOLS
Code Generator
SPONSOR
Get an all-in-one Marketing Platform
Know the tastes of each customer. Connect with fans based on how they use your app.
REGULAR EXPRESSION
1 match (~0ms)
/ ^T / gi
TEST STRING
SWITCH TO UNIT TESTS
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m
Troceamos todos los ingredientes indicados en la proporción que os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de agua de la nevera y 50 ml de vinagre de Jerez, triturando todo en la batidora de vaso o Thermomix. Una vez triturado, pasamos el gazpacho resultante por el colador fino y lo metemos en la nevera un par de horas para que enfríe bien.
SUBSTITUTION

Ejemplo - PHP


regex101.com
regular expressions 101
SAVE & SHARE
Save Regex ctrl+s
FLAVOR
PCRE (PHP)
ECMAScript (JavaScript)
Python
Golang
TOOLS
Code Generator
Regex Debugger
SPONSOR
Get an all-in-one Marketing Platform
Know the tastes of each customer. Connect with fans
REGULAR EXPRESSION
1 match,
/ ^T
TEST STRING
SWITCH TO
Tomate pera: 1 kg
Pimiento verde italiano: 1
Pepino: 1
Dientes de ajo: 2
Aceite de oliva virgen: 50 ml
Pan de hogaza duro: 50 g
Agua: 250 ml
Sal: 5 g
Vinagre de Jerez: 30 ml
Tiempo total: 15 m
Troceamos todos los ingredientes indicados en la proporción
os he puesto y añadimos 50 ml de aceite de oliva, 250 ml de
de la nevera y 50 ml de vinagre de Jerez, triturando todo en
batidora de vaso o Thermomix. Una vez triturado, pasamos el
gazpacho resultante por el colador fino y lo metemos en la
nevera un par de horas para que enfríe bien.
SUBSTITUTION
REGEX FLAGS
global
Don't return after first match
multi line
^ and $ match start/end of line
insensitive
Case insensitive match
extended
Ignore whitespace
eXtra
Disallow meaningless escapes
single line
Dot matches newline
unicode
Match with full unicode
Ungreedy
Make quantifiers lazy
Anchored
Anchor to start of pattern
Jchanged
Allow duplicate subpattern

Ejemplo - Python



Contenidos

1. Introducción
2. Sintaxis de expresiones regulares
3. **Ejemplos prácticos**
   - **HTML5**
   - **JavaScript**
   - **Casos de uso**

HTML5

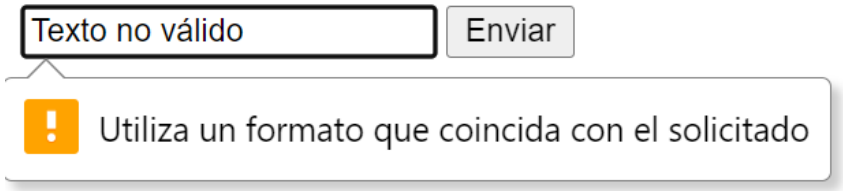
- HTML5 incluye un atributo **`pattern`** para los elementos de tipo input, que permite validar el valor introducido por el usuario, comparándolo con una expresión regular:

```
<input type="text" pattern="REGEX"/>
```

- Ejemplo:

```
<input type="text" pattern="\d{5}"/>
```

- **Por sí solo no es buena UX**, ya que no le indica al usuario el formato esperado:



En JavaScript

- Las expresiones regulares en JavaScript pueden declararse de dos formas distintas:

```
function validate() {
    //Notación literal: se indica mediante barras
    let re = /\d{5}/gm;

    //Mediante constructor: no se incluyen barras, sino comillas
    //Los modificadores se incluyen en el segundo parámetro
    let re = new RegExp("\d{5}", "gm");
}
```

- Documentación:
  - https://www.w3schools.com/jsref/jsref_obj_regexp.asp
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp

Uso de expresiones regulares en JavaScript

- Métodos de la clase **String**
  - `cadena.match(regex)`: devuelve *null* si no encuentra el patrón definido por *regex* dentro de la cadena; si lo encuentra, devuelve un array con información sobre la coincidencia (*input, index, lastIndex*).
  - `cadena.search(regex)`: devuelve -1 si no encuentra el patrón definido por *regex* dentro de la cadena; si lo encuentra, devuelve el índice en el que empieza la coincidencia.
  - `cadena.replace(regex, texto)`: devuelve una copia de la cadena en la que las coincidencias del patrón definido por *regex* han sido reemplazadas por el *texto* indicado.
  - Si la opción *g* está activa, reemplaza todas las coincidencias, si no, sólo la primera.

Uso de expresiones regulares en JavaScript (II)

- Métodos de la clase **RegEx**
  - `regex.test(cadena)`: devuelve *true* si encuentra alguna coincidencia del patrón dentro de la cadena o *false* en caso contrario.
  - `regex.exec(cadena)`: devuelve *null* si no encuentra el patrón dentro de la cadena; si lo encuentra, devuelve un array con información sobre la coincidencia (*input, index, lastIndex*).

Equivalencias de métodos

- Los métodos de expresiones regulares de **String** y de **RegExp** son equivalentes:
  - `regex.exec(cadena) === cadena.match(regex)`
  - `regex.test(cadena) === (cadena.search(regex) !== -1)`

- El uso en otros lenguajes es muy parecido, por ejemplo, Python tiene los métodos `re.search(), re.match()` y `re.findall()` en el modulo nativo `re`.

Casos de uso: validación de formato de cadenas

- Validación del formato básico de un DNI (sin tener en cuenta si la letra es correcta):

```
function isValidDNI(dni) {
    let regex_dni = /\d{8}[A-Z]/;
    return regex_dni.test(dni);
}
```

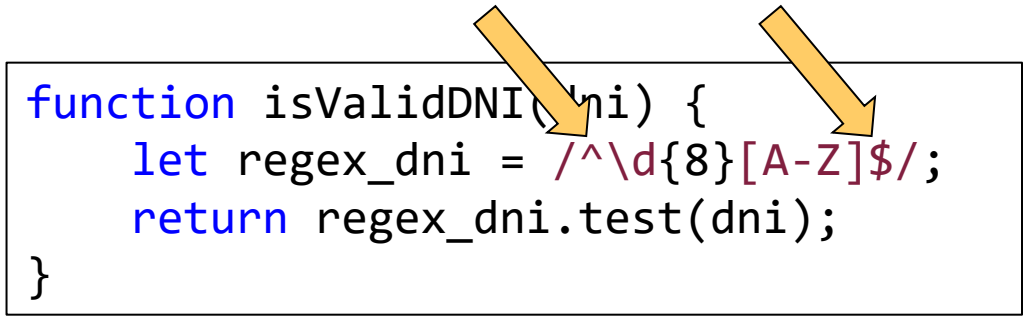
Casos de uso: validación de formato de cadenas

- Validación del formato básico de un DNI (sin tener en cuenta si la letra es correcta):
- Ojo: en JS, todos los métodos de expresiones regulares buscan la expresión regular **en cualquier parte de la cadena**
  - La cadena `"!+-_12345678A....."` sería considerada válida

```
function isValidDNI(dni) {
    let regex_dni = /\d{8}[A-Z]/;
    return regex_dni.test(dni);
}
```

Casos de uso: validación de formato de cadenas

- Validación del formato básico de un DNI (sin tener en cuenta si la letra es correcta):
- Ojo: en JS, todos los métodos de expresiones regulares buscan la expresión regular **en cualquier parte de la cadena**
  - La cadena `"!+-_12345678A....."` sería considerada válida
  - Solución: usar los meta-caracteres **^** y **$** para indicar principio y final de la cadena



```
function isValidDNI(dni) {
    let regex_dni = /^\d{8}[A-Z]$/;
    return regex_dni.test(dni);
}
```

Casos de uso: extracción de información

- Objetivo: extraer el día, mes y año en una fecha con formato “d/m/a” donde tanto el día como el mes pueden tener uno o dos caracteres.
  - Los grupos de captura son útiles en este caso:

Casos de uso: extracción de información

- Objetivo: extraer el día, mes y año en una fecha con formato “d/m/a” donde tanto el día como el mes pueden tener uno o dos caracteres.

```
function getDateInfo(date_string) {
    let re_date = /(\d{1,2})\/(\d{1,2})\/(\d{4})/;

    let match = date_string.match(re_date);
    if (match !== null) {
        console.log("The day is:", match[1]);
        console.log("The month is:", match[2]);
        console.log("The year is:", match[3]);
    }
}
```

Casos de uso: extracción de información

- Objetivo: extraer el día, mes y año en una fecha con formato “d/m/a” donde tanto el día como el mes pueden tener uno o dos caracteres.

```
function getDateInfo(date_string) {
    let re_date = /(\d{1,2})\/(\d{1,2})\/(\d{4})/;

    let match = date_string.match(re_date);
    if (match !== null) {
        console.log("The day is:", match[1]);
        conso                          h[2]);
        conso                          [3]);
    }
}
```

Si la cadena coincide con el patron, “match” será un array con los grupos de captura

Casos de uso: extracción de información

- Objetivo: extraer el día, mes y año en una fecha con formato “d/m/a” donde tanto el día como el mes pueden tener uno o dos caracteres.

```
function getDateInfo(date_string) {
    let re_date = /(\d{1,2})\/(\d{1,2})\/(\d{4})/;

    let match = date_string.match(re_date);
    if (match !== null) {
        console.log("The day is:", match[1]);
        console.log("The month is:", match[2]);
        console.log("The year is
    }
}
```

Los grupos aparecen por orden de definición en la RegEx. La posición 0 está siempre reservada para la coincidencia con la expresión completa.

Casos de uso: extracción + validación

- Objetivo: comprobar si una cadena representa una dirección IPv4 válida (4 números 0-255 separados por puntos)
- Idea: capturar cada grupo de números y validarlo numéricamente

```
/^(\d+)\.(\d+)\.(\d+)\.(\d+)$/
```

Casos de uso: extracción + validación

- Objetivo: comprobar si una cadena representa una dirección IPv4 válida (4 números 0-255 separados por puntos)

```
function isValidIP(addr_string) {
    let re_address = /^(\d+)\.(\d+)\.(\d+)\.(\d+)$/;
    let match = addr_string.match(re_address);

    if (match === null) {
        return false;
    }

    return match.slice(1).every(validIPnumber);
}

function validIPnumber(s) {
    let n = parseInt(s);
    return n >= 0 && n <= 255;
}
```

Casos de uso: extracción + validación

- Conclusión: no siempre es necesario hacer **todo** el proceso con expresiones regulares, especialmente para números
  - Se puede, pero...

```
/^([01]?[0-9]?[0-9]|2[0-4][0-
9]|25[0-5]).([01]?[0-9]?[0-9]|2[0-
4][0-9]|25[0-5]).([01]?[0-9]?[0-
9]|2[0-4][0-9]|25[0-5]).([01]?[0-
9]?[0-9]|2[0-4][0-9]|25[0-5])$/
```

Some people, when confronted with a problem, think, "I know, I'll use regular expressions." Now they have two problems.

Jaime Zawinski
12 Aug, 1997

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta

```
17/May/2022 19:10:39 | [WEB] GET /js/libs/jquery-3.4.1.min.js from 127.0.0.1 - 304
17/May/2022 19:10:39 | [WEB] GET /js/libs/bootstrap.min.js from 127.0.0.1 - 304
17/May/2022 19:10:39 | [WEB] GET /js/darkmode.js from 127.0.0.1 - 304
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
17/May/2022 19:10:39 | [WEB] GET /js/utils/session.js from 127.0.0.1 - 304
17/May/2022 19:10:39 | [WEB] GET /js/api/_usuarios.js from 127.0.0.1 - 304
17/May/2022 19:10:39 | [WEB] GET /js/api/common.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /js/renderers/gallery.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /js/renderers/messages.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /js/api/_usuariosyfotos.js from 127.0.0.1 - 200
17/May/2022 19:10:40 | [WEB] GET /js/utils/parseHTML.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /js/renderers/photos.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /js/api/usuariosyfotos.js from 127.0.0.1 - 304
17/May/2022 19:10:40 | [WEB] GET /nav_bar.html from 127.0.0.1 - 304
17/May/2022 19:10:40 | [API] GET /api/v1/usuariosyfotos from 127.0.0.1 - 200
17/May/2022 19:10:40 | [API] GET /api/v1/usuarios/1 from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

/ /

```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+)                                    /
```

El método son letras mayúsculas

```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+)                                    /
```

Un espacio

```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+) (.*)                    /
```

Podemos generalizar la URL como “cualquier cadena”

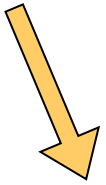
```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+) (.*) from                    /
```

Cadena literal “ from ”



```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+) (.*) from (.*)          /
```

(cualquier cadena)

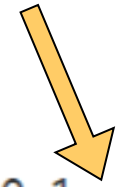
```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+) (.*) from (.*) -      /
```

Cadena literal " - "



```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

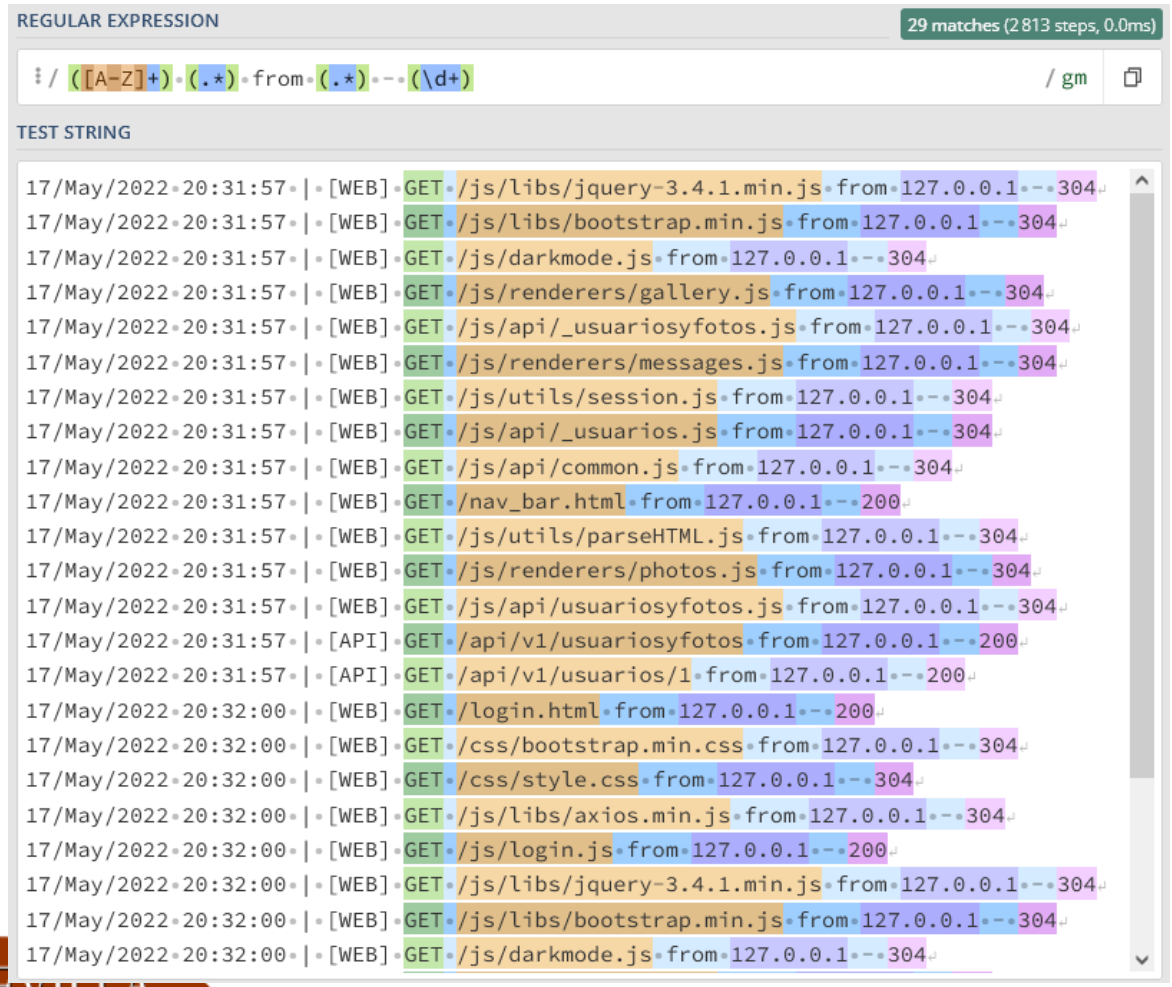
- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- En muchos casos reales, para extraer información estructurada usando RegEx, basta con analizar el formato de la cadena y usar elementos más sencillos y/o constantes como referencias

```
/([A-Z]+) (.*) from (.*) - (\d+)/
```


```
17/May/2022 19:10:39 | [WEB] GET /js/index.js from 127.0.0.1 - 200
```

Casos de uso: extracción de información (II)

- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta

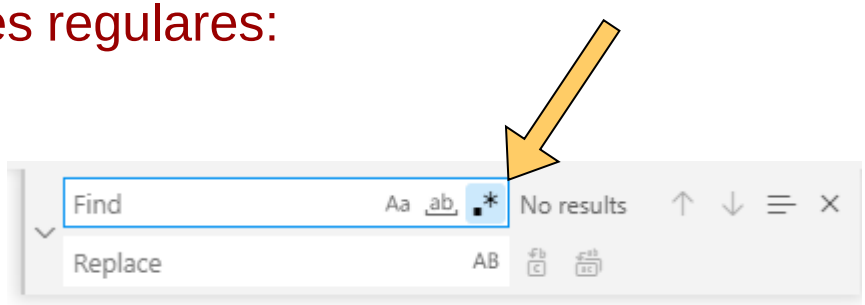


Casos de uso: extracción de información (II)

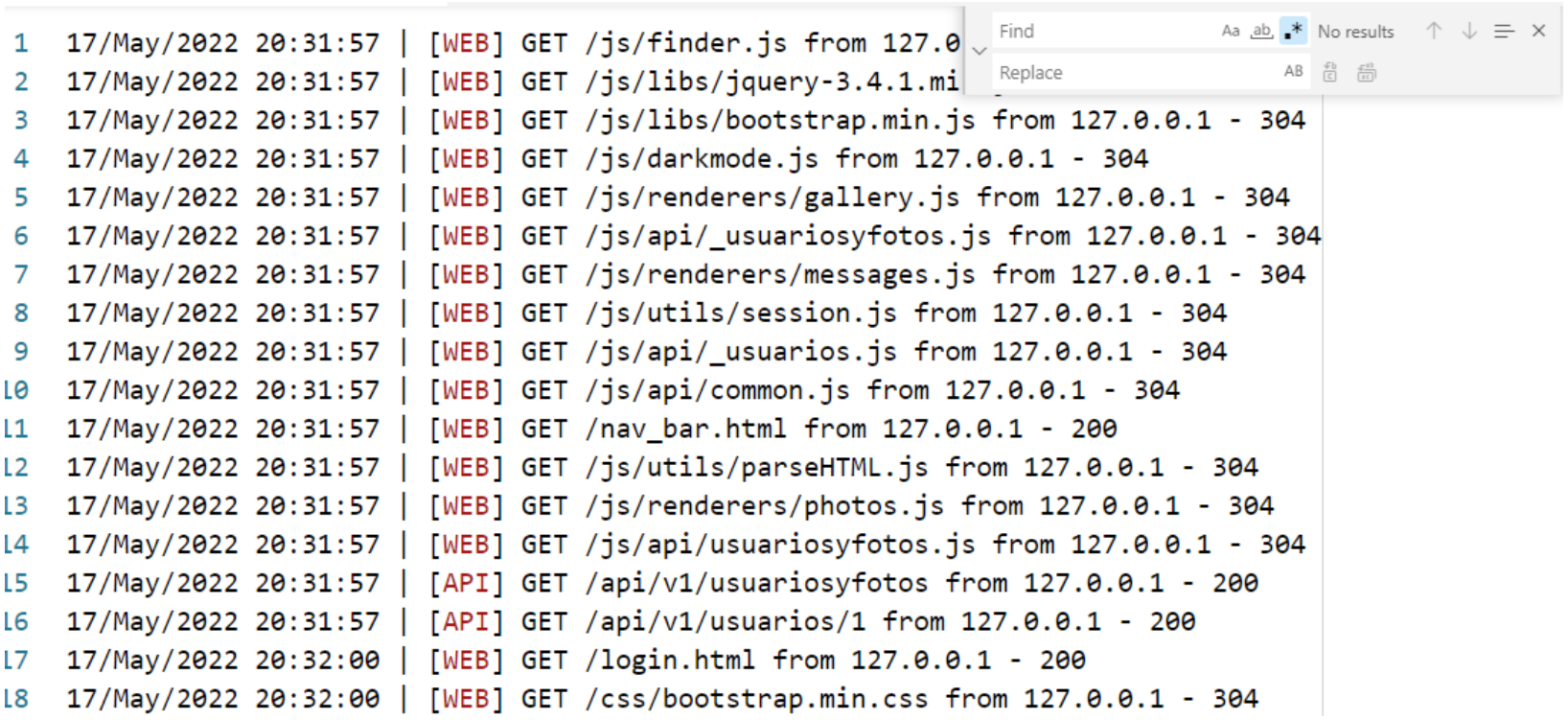
- Objetivo: extraer la siguiente información de cada línea de log de Silence: método HTTP, ruta, dirección IP y código de respuesta
- Conclusión: en la práctica, no es necesario escribir la expresión regular “perfecta” para extraer información de una cadena
  - Basta con que sea lo bastante específica para la cadena en cuestión
  - Hay que tener cuidado si el formato puede cambiar en el futuro

Casos de uso: reemplazos en VS Code

- Podemos reemplazar todo un patron con un texto fijo, o usar los grupos de captura dentro de una expression
  - Por ejemplo, en VS Code se denotan con $N, donde N es el número del grupo (empezando en 1)
  - Pulsamos CTRL+H (reemplazar) y activamos el modo expresiones regulares:



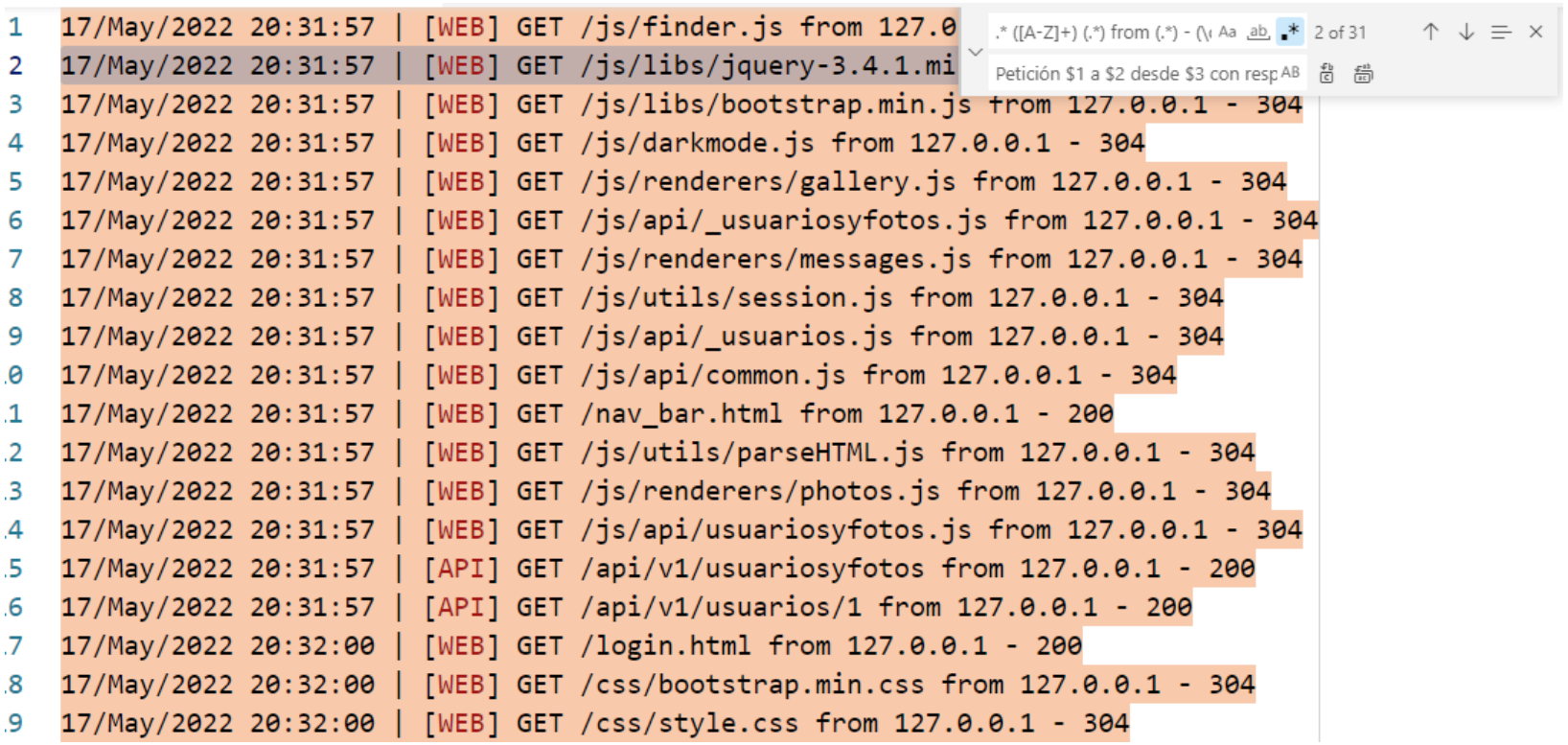
Casos de uso: reemplazos en VS Code



Buscar: `.* ([A-Z]+) (.*) from (.*) - (\d+)`

Reemplazar: `Petición $1 a $2 desde $3 con respuesta $4`

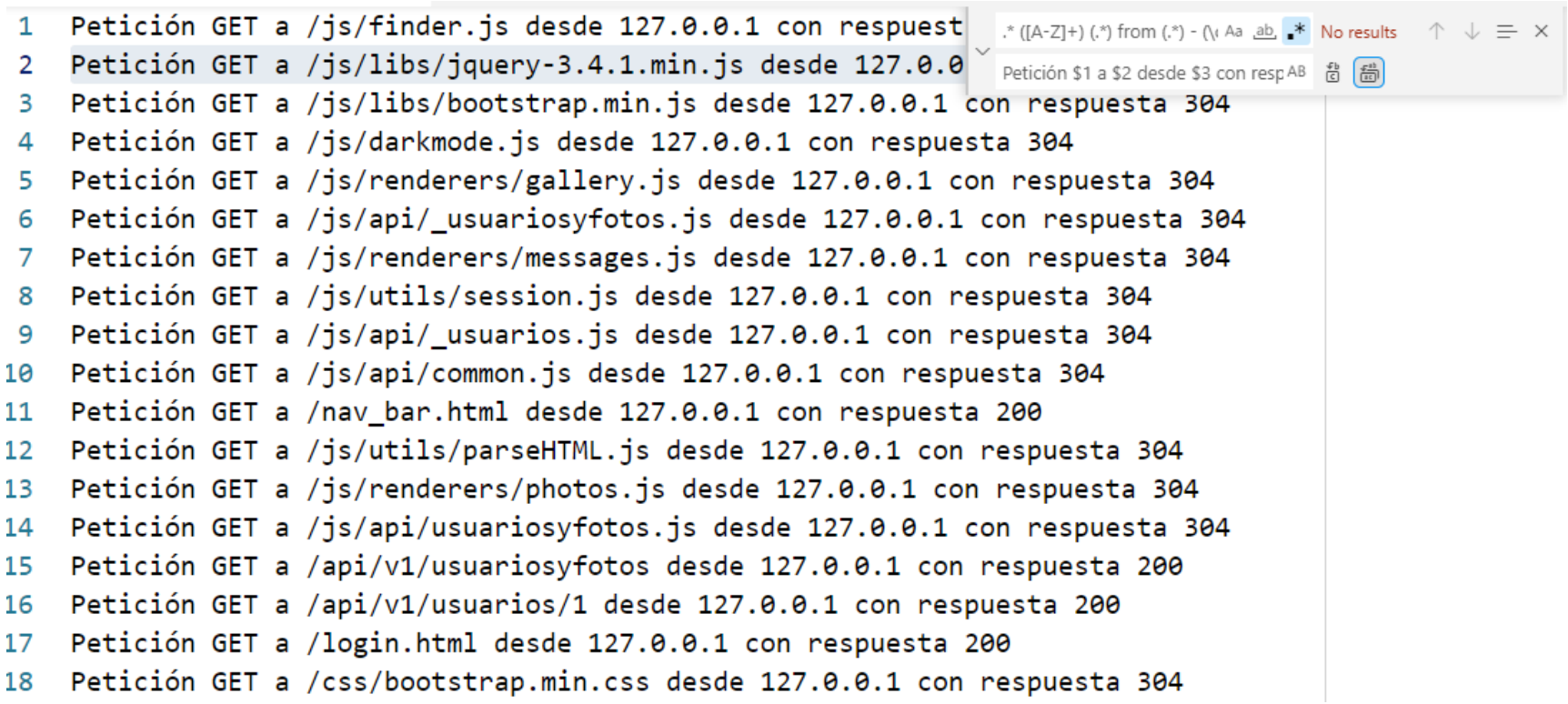
Casos de uso: reemplazos en VS Code



Buscar: `.* ([A-Z]+) (.*) from (.*) - (\d+)`

Reemplazar: `Petición $1 a $2 desde $3 con respuesta $4`

Casos de uso: reemplazos en VS Code



Buscar: `.* ([A-Z]+) (.*) from (.*) - (\d+)`

Reemplazar: `Petición $1 a $2 desde $3 con respuesta $4`

Conclusiones

- Las expresiones regulares son una herramienta muy poderosa para:
  - Validar formatos de cadenas
  - Extraer información
  - Realizar reemplazos complejos
- Aunque no siempre son la herramienta más adecuada, especialmente para validaciones más complejas y restricciones numéricas
- Es muy aconsejable usar herramientas online a la hora de crear RegEx desde cero para poder ver si funcionan y/o tienen errores de sintaxis.

Recursos de interés

- RegEx avanzado: lookarounds
  - https://www.rexegg.com/regex-lookarounds.html
- Crucigramas para practicar RegEx
  - https://regexcrossword.com/
- iHateRegex: colección de expresiones regulares y motor interactivo
  - https://ihateregex.io/
- Por qué no se debe usar RegEx para parsear HTML
  - https://stackoverflow.com/questions/1732348/regex-match-open-tags-except-xhtml-self-contained-tags/1732454#1732454

Expresiones regulares (RegEx)

Departamento de Lenguajes y Sistemas Informáticos

Universidad de Sevilla

Daniel Ayala, Carlos Arévalo, José Calderón, Margarita Cruz, Inma Hernández, David Ruiz