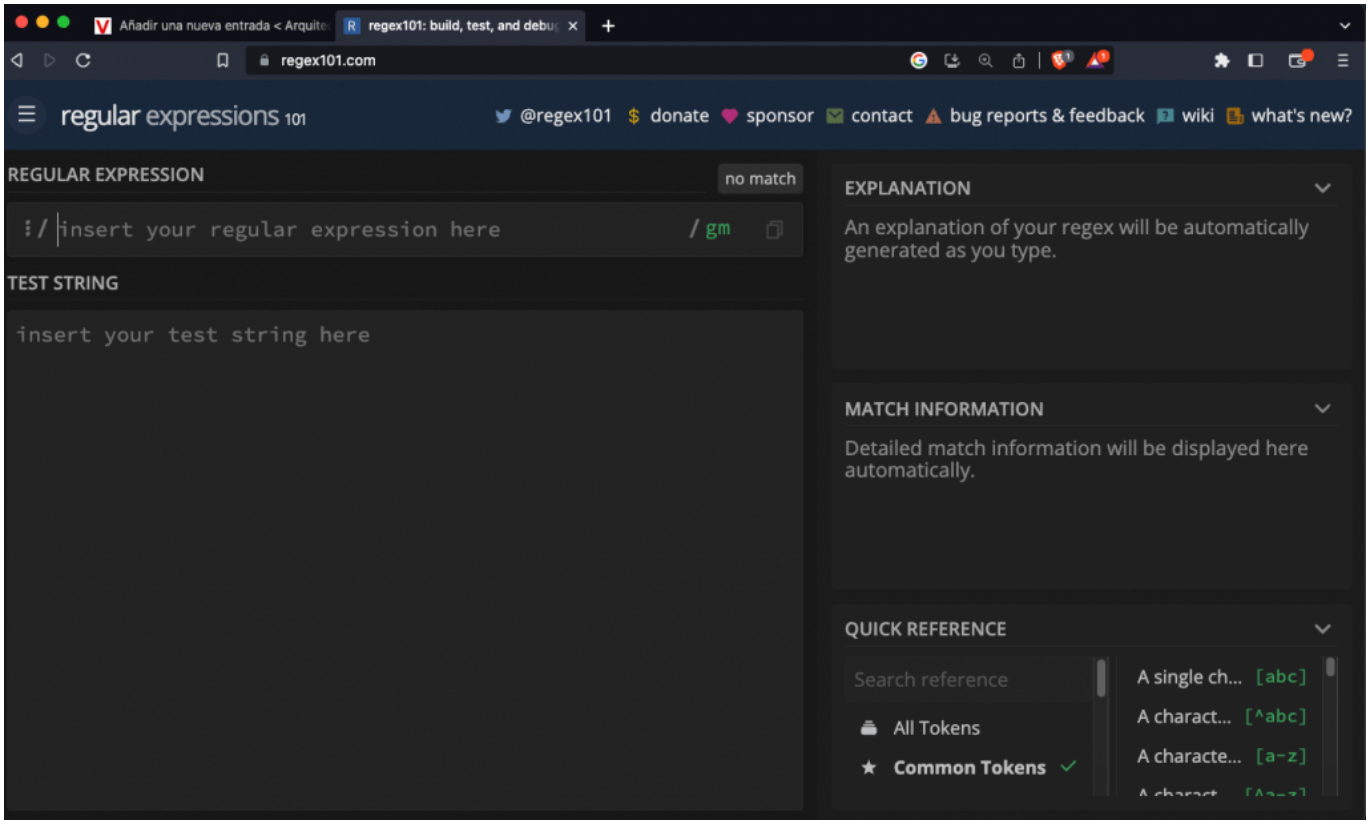
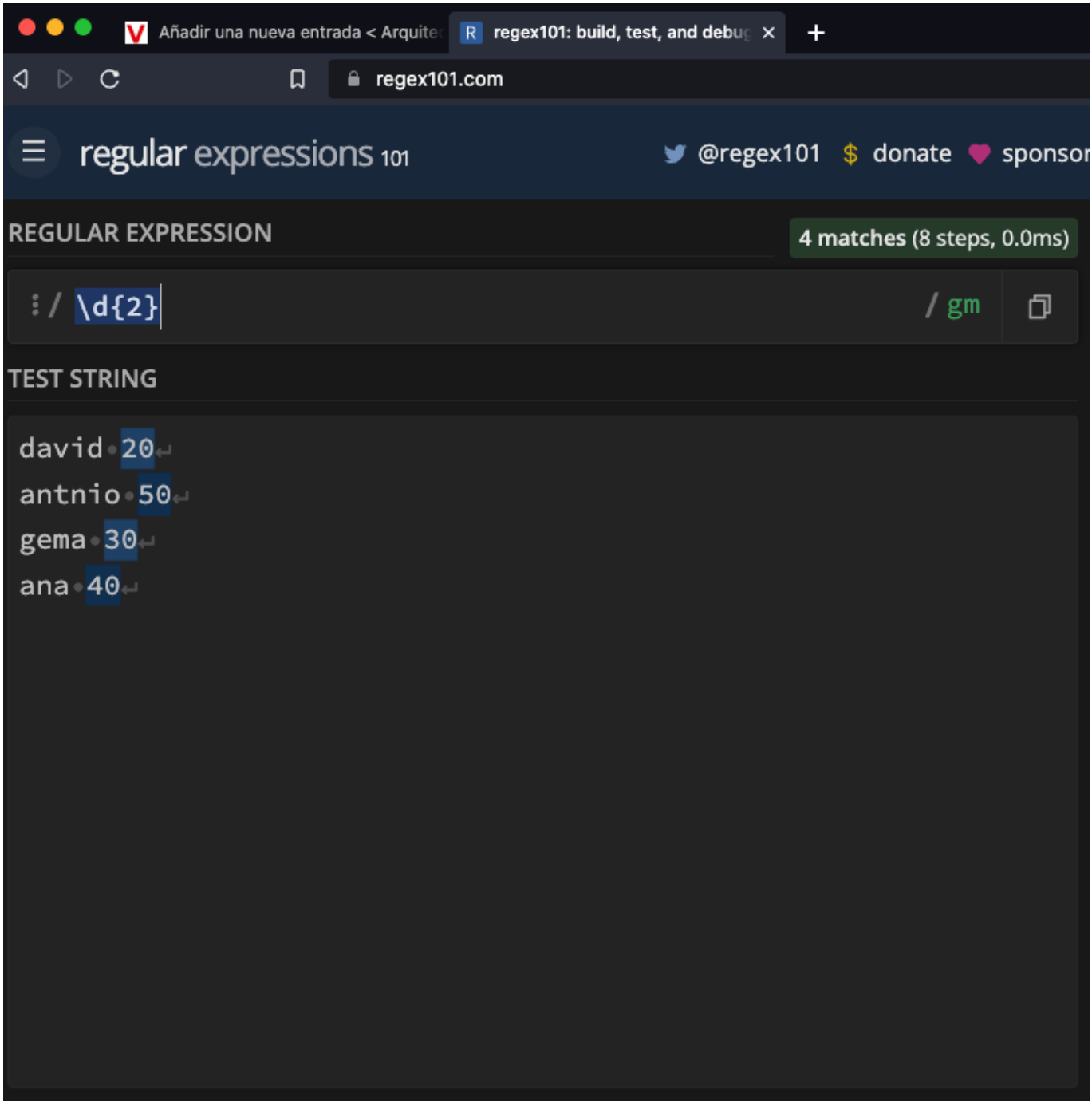
El manejo de Java Regular Expression en nuestro código es muy muy común. Las expresiones regulares se usan en el día a día para validar patrones de cadenas de forma rápida. Una de las webs que más utilizo para validar las expresiones regulares es 101 Regex.



Esta web nos permite de forma sencilla escribir un texto y asociar una expresión regular a él de tal forma que la valida sobre la marcha y nos muestra que subcadena o conjuntos de subcadenas coinciden con la. expresión regular. Vamos a ver un ejemplo sencillo.



En este caso hemos intentado buscar en el texto:

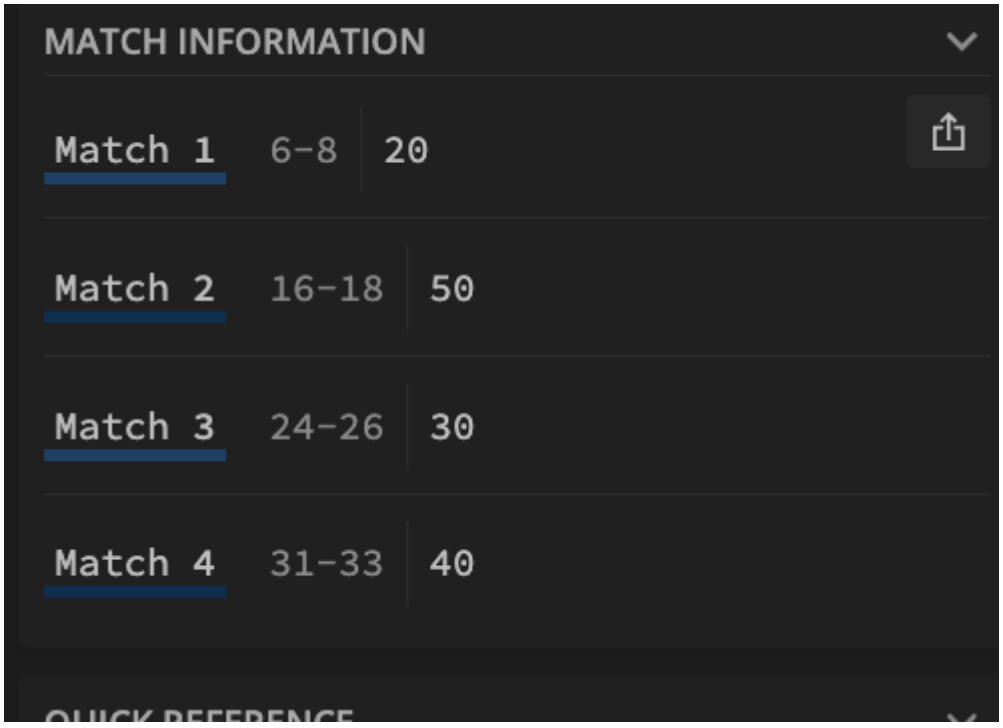
```
david 20
antnio 50
gema 30
```

```
ana 40
```

Las subcadenas que sean dígitos usando como expresión regular \d{2} que significa que tenemos que buscar dos dígitos exactamente.

Regular Expression y coincidencias

Como se puede ver tenemos en el menú de la derecha donde se han encontrado las coincidencias en nuestro texto.



Es momento de probar la expresión regular en nuestro código de Java y ver como es capaz de buscar coincidencias en una cadena e imprimirlas por la consola.

Java Regular Expression

```
package com.arquitecturajava;

import java.util.regex.Matcher;
```

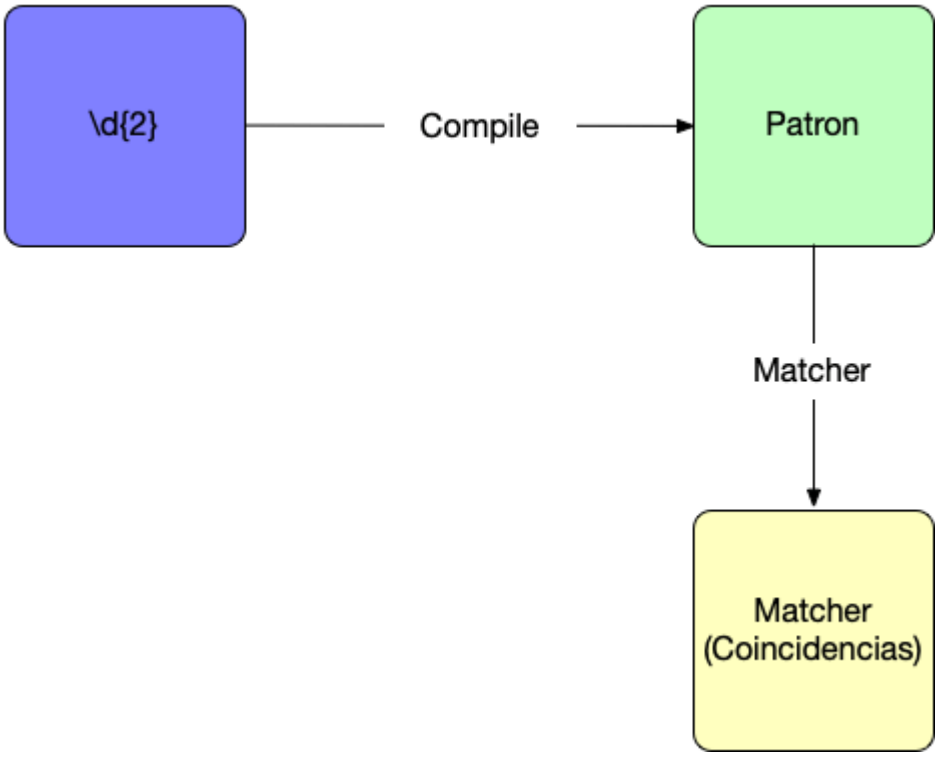
```
import java.util.regex.Pattern;

public class Principal {

        public static void main(String[] args) {
                String texto="antonio 20  david 30 ana 40 gema 50";
                Pattern patron = Pattern.compile("\\d{2}",
Pattern.CASE_INSENSITIVE);
                Matcher m= patron.matcher(texto);
                  while(m.find()) {
                        System.out.println(m.group());
                  }
        }

}
```

En este caso hemos usado la clase Pattern que define un patrón . Este patrón es el de doble dígito \d{2} lo que sucede es que el carácter \ es un carácter especial en Java y hay que escaparlo . Por ello lo que hemos hecho es poner una doble \ . Realizada esta operación se compila el patrón para poder usarlo.



Aplicamos el método matcher y le pasamos el texto. Esto nos devolverá un matcher que se parece a un iterador y permite recorrer todas las coincidencias a través del método find:

```
20
30
40
50
```

Acabamos de ver como realizar un manejo sencillo de Java Regular Expression buscando un conjunto de coincidencias dentro de un texto determinado.

Otros artículos relacionados

- TypeScript Class Expression y simplicidad
- Spring Expression Language
- Java List to Map y el uso de Collectors
- El patron Repository y la explosión de métodos
- El patrón MVC , arquitectura cliente vs servidor